FRACTURE CHARACTERIZATION AND ESTIMATION OF FRACTURE POROSITY OF NATURALLY FRACTURED RESERVOIRS WITH NO MATRIX POROSITY USING STOCHASTIC FRACTAL MODELS

A Dissertation

by

TAE HYUNG KIM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2007

Major Subject: Petroleum Engineering

FRACTURE CHARACTERIZATION AND ESTIMATION OF FRACTURE POROSITY OF NATURALLY FRACTURED RESERVOIRS WITH NO MATRIX POROSITY USING STOCHASTIC FRACTAL MODELS

A Dissertation

by

TAE HYUNG KIM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	David S. Schechter
Committee Members,	Thomas A. Blasingame
	John Keyser
	Robert A. Wattenbarger
Head of Department,	Stephen A. Holditch

December 2007

Major Subject: Petroleum Engineering

ABSTRACT

Fracture Characterization and Estimation of Fracture Porosity of Naturally Fractured
Reservoirs with No Matrix Porosity Using Stochastic Fractal Models. (December 2007)

Tae Hyung Kim, B.S., Inha University, Korea;
M.S., Seoul National University, Korea
Chair of Advisory Committee: Dr. David S. Schechter

Determining fracture characteristics at the laboratory scale is a major challenge. It is known that fracture characteristics are scale dependent; as such, the minimum sample size should be deduced in order to scale to reservoir dimensions. The main factor affecting mechanical and hydrological characteristics of natural fractures is aperture distribution, which is a function of scale and confining pressure, rather than roughness of one fracture surface. Scale and pressure dependencies of artificial and natural fractures were investigated in this study using an X-Ray CT Scanner. Fractal dimension, *D*, and amplitude parameter, *A*, of fracture aperture approaches a constant value with increased sampling area, similar to the behavior of fracture roughness. In addition, both parameters differ under different confining pressures for a reference sampling area. Mechanical properties of fracture-fracture deformation behavior and fracture normal stiffness were obtained from CT scan data as well.

Matrix porosity is relatively easy to measure and estimate compared to fracture porosity. On the other hand, fracture porosity is highly heterogeneous and very difficult to measure and estimate. When matrix porosity of naturally fractured reservoirs (NFR) is negligible, it is very important to know fracture porosity to evaluate reservoir performance. Since fracture porosity is highly uncertain, fractal discrete fractal network (FDFN) generation codes were developed to estimate fracture porosity. To reflect scale dependent characteristics of fracture networks, fractal theories are adopted. FDFN modeling technique enables the systematic use of data obtained from image log and core analysis for estimating fracture porosity. As a result, each fracture has its own fracture aperture distribution, so that generated FDFN are similar to actual fracture systems. The results of this research will contribute to properly evaluating the fracture porosity of NFR where matrix porosity is negligible.

DEDICATION

My graduate studies at Texas A&M University would not have been possible without the help from God and prayer of my parents and sister.

This dissertation is dedicated to:

my father,	*Chang Ok Kim,*
my mother,	*Chung Ae Ryu,*
my sister:	*Bo Youn Kim;*

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

LIST OF TABLES

TABLE Page

CHAPTER I

INTRODUCTION

Naturally fractured reservoirs (NFR) are classified into four types (Nelson). Among the four types[1] **(Fig. 1.1)**, there are: Type 1-fractures provide the essential storage capacity and permeability in a reservoir. The matrix has little porosity or permeability. Type 2-rock matrix provides the essential storage capacity and fractures provide the essential permeability in a reservoir. The rock matrix has low permeability, but may have low, moderate, or even high porosity. Type 3-fractures provide a permeability assist in an already economically producible reservoir that has good matrix porosity and permeability. Type 4-fractures do not provide significant additional storage capacity or permeability in an already producible reservoir, but instead create anisotropy. Type 1 NFR have highly heterogeneous and scale dependant characteristics so that fracture porosity measurement is almost impossible through logging or core analysis. Thus, fracture porosity remains uncertain and the economic feasibility of Type 1 NFR is always questionable.

Some researchers have studied the characteristics of a single fracture (on a small scale)[2-10], and others have studied fracture networks (on a large scale)[11-27]. However, results of these research studies are not integrated for estimating fracture porosity. As a result, considerable information exists around the industry, but few can answer the question: What Type 1 fracture porosity is NFR?

In order to calculate fracture porosity or fracture volume, two kinds of information are needed, fracture length or area and aperture. Outcrop maps are used to obtain fracture length distribution information based on an analog outcrop. Fracture area is deduced from length distribution using mathematical means[28]. Fractal dimension of fracture center distribution is another parameter obtained from outcrop maps. Fracture aperture distribution is obtained from electrical borehole scans[29] or lab scale experiments[2, 4, 8, 9].

This dissertation follows the style of *SPE Journal*.

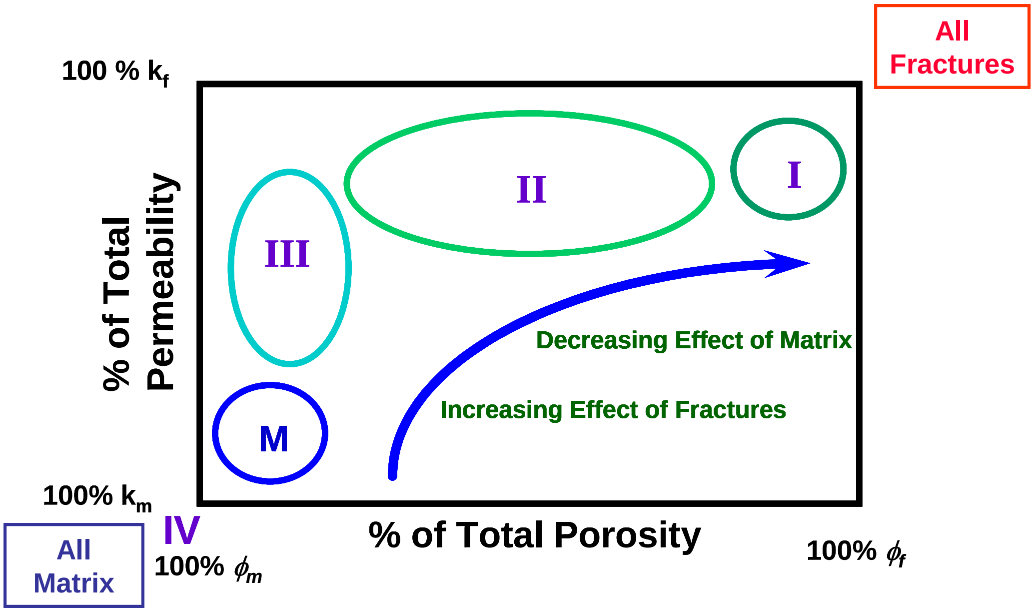


Fig. 1.1-Schematic distribution of naturally fractured reservoirs (after Nelson[1]).

Naturally occurring fractures serve as main storage and main conduits for hydrocarbon flow in Type 1 NFR. Therefore, it is crucial to characterize fractures and fracture networks to estimate storage capacity and to understand physical phenomena concerning fluid flow through fractures. Fracture characterization is mainly performed at the laboratory scale rather than the field scale due to high costs and difficulties in controlling test conditions. Thus, improvement of rough surface fracture characterization represents an important subject for many areas of study.

It is well known that hydrological and mechanical properties of rock fractures depend on the scale used. Lanardo *et al.*[6] suggested the concept of stationary threshold. For sampling window sizes larger than the stationary threshold, parameters such as fractal dimension remain almost constant. Using a 3-D laser scanner, they obtained geometric information about a fracture surface, and then established the stationary threshold using standard deviation. Fardin *et al.*[7] confirmed the validity of the concept of stationary threshold. However, they suggested that fractal dimension is a more appropriate criterion than standard deviation for deciding stationary threshold. The stationary threshold of fracture aperture, which is the result of the interaction of two rough surfaces found on fracture faces, has not yet been studied.

Myer[3] concluded that hydrological, mechanical and seismic properties of fractures are dominated by the geometry of the void space within a fracture. Several studies identifying aperture distribution have been conducted. Pyrak-Nolte *et al.*[2] used Wood's-metal injection techniques and showed that aperture distribution is affected by normal stress. The limitation of this method is that fracture must be separated, and removing injected metal can damage fracture surfaces. Keller[4], Bertels and DiCarlo[8], and Muralidharan *et al.*[9] used an X-Ray CT Scanner to measure aperture distribution. The advantage of this technique is that fracture aperture can be measured without damaging specimens and the aperture geometry is preserved because the two rock halves do not need to be separated before the end of the experiments. Gale[30], Hakami and Larsson[5], and Konzuk and Kueper[10] measured aperture by analyzing digital images of fracture traces. Dyed resin was injected into fracture after flow experiments, then the core sample containing the fracture was cut into several sections normal to the fracture surface. Following that, traces of fracture were imaged. This method enables one to analyze fracture aperture after flow experiments, but inevitably destroys specimens.

Since hydrological and mechanical properties of fracture depend on fracture aperture distribution, which depends on scale and pressure rather than on geometry of one fracture surface, proper study requires simultaneous characterization of fracture aperture and determination of scale and pressure dependencies. To our knowledge, studies on scale and pressure dependent properties of aperture have not been conducted so far. The main objective of this research is to determine fracture representative properties depending on scale and pressure in a laboratory scale using an X-Ray CT Scanner. In addition, we examine the ability of the X-Ray CT Scanner to measure mechanical properties of fracture.

The heterogeneous nature of fracture patterns makes it impossible to use deterministic methods in the estimation of fracture porosity. Thus, stochastic methods are adopted to generate discrete fracture networks (DFN)[31-33]. DFN studies have not fully utilized the results of field and lab measurements. Almost all DFN studies estimate fracture porosity based on the parallel plate assumption. In addition, DFN studies have not thoroughly considered the scale dependent properties of fracture porosity. Therefore, current DFN studies are not helpful in reducing the uncertainty in fracture porosity.

The key to reducing the uncertainty is utilizing available data as much as possible and generating more realistic fracture networks. To that end, the study of the FDFN generation concentrates on two objectives: generating real-life fracture networks and assembling fracture networks with more realistic aperture distributions. The key to achieving the former is adopting fractal theories. The main advantage of fractals is that fractals are independent of scale, which allows us to utilize any data regardless of the measurement scale. Fracture aperture distribution is also known to have fractal characteristics, so the latter objective can be reached by applying fractals. To mimic real fracture networks, each fracture has its own roughness rather than a single value as utilized in the parallel plate method.

2D and 3D fractal discrete fracture network (FDFN) generation codes were developed and verified based on fractal theories. In addition, effective means of fracture porosity estimation are suggested. The suggested method of fracture porosity estimation uses the advantages of 2D and 3D generation.

CHAPTER II

ANALYSIS OF ARTIFICIAL FRACTURE APERTURE

2.1 Experimental Procedures of Measuring Aperture Using X-Ray CT Scanner

2.1.1 Calibration

The X-Ray CT Scanner used is Universal Systems HD-350E **(Fig. 2.1)**, which scans as fast as 1 sec/scan. It can analyze a core plug up to 81 cm long and gives a spatial resolution of 0.35 mm. In this study, the data analyzed are raw pixel data rather than the spatial data from CT image. If CT images (digital images) are analyzed directly to calculate aperture, accuracy of 0.35 mm spatial resolution will be questionable. According to Shannon’s sampling theorem, the digitizing device must utilize a sampling interval that is no greater than one-half size of the smallest resolvable feature of the optic image. Thus, sampling should be done less than 125 μm for 0.35 mm spatial resolution. X-Ray CT Scanner oversamples and oversampled data provide extra pixels that do not theoretically contribute to the spatial resolution of a digital image, thus they can improve the accuracy of the scan feature and aperture size smaller than a spatial resolution is able to be calculated.

Fig. 2.1-Universal Systems HD-350E.

The X-Ray CT Scanner identifies the density differences between various objects[34]; the density of an air-filled fracture is less than the density of the rock. Thus CT data of fractured rock can differentiate matrix from fracture. Because CT numbers are merely functions of densities, a calibration technique must be applied to correlate CT numbers with actual aperture size[4, 8, 9]. It was found that fracture area is represented by low CT numbers, and the area increases as aperture size increases[9]. Some may think that CT measurements of air-filled fractures are similar to that of the air (-1000), but this is only true for air in a large void space, rather than a narrow space such as fracture. The CT number for an air-filled fracture is affected by surrounding rock, which is usually called "oversampling"[9]. Thus, distinct calibrations are required for different rocks.

The experimental setup consists of the X-Ray CT Scanner, two halves of a rock specimen, feeler gauges, and a core holder. The procedures for calibration follow:

(1) The rock specimen was obtained from an intact part of a Spraberry sandstone core and was cut using a diamond saw along the longitudinal direction.

(2) The surfaces were grinded and smoothed to reduce roughness as much as possible.

(3) Various known sizes of feeler gauges (52, 76, 127, 203, 279, 330 μm) were inserted between the halves.

(4) The feeler-gauges-inserted specimen was placed in the core holder, then 3.4 MPa confining pressure was applied to insure firm contact between the gauges and the specimen.

(5) Multiple CT scans were taken between the feeler gauges.

Fig. 2.2 shows the increase of the distance between the two halves with increasing feeler gauge size. The CT numbers in the fracture area were integrated **(Fig. 2.3)**; the integrated CT signal (*IntCT*) values under different sizes of feeler gauges were then plotted along with aperture sizes **(Fig. 2.4)**. The plot shows a linear relationship. The correlation of aperture derived from this calibration curve for Spraberry sandstone is expressed by Eq. 2.1:

$$Fracture\ Aperture[\mu m] = \frac{IntCT - 42.236}{5.866} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

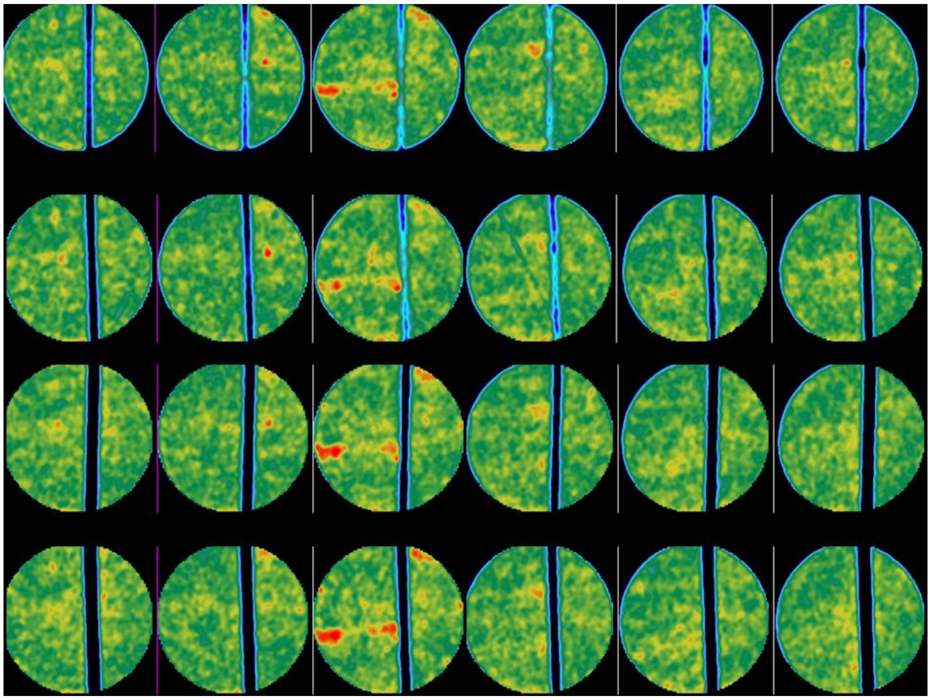
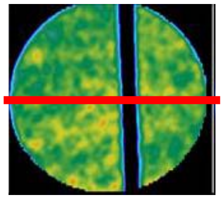
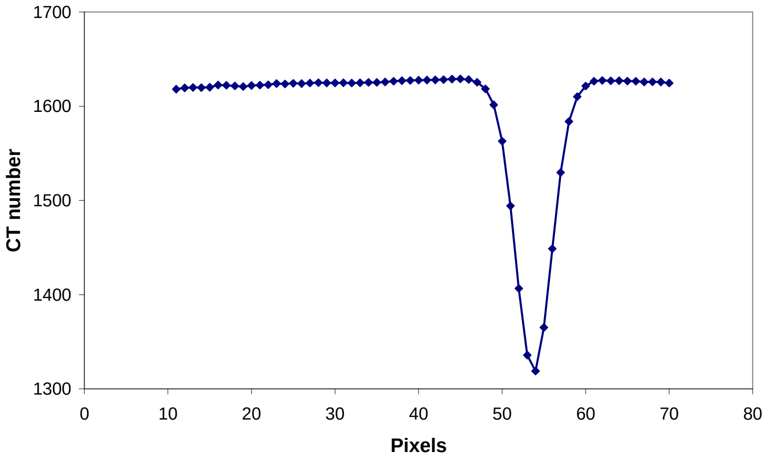


Fig. 2.2-CT images with different sizes of feeler gauges. From the top line, sizes of feeler gauges are 51, 127, 279, and 330 μm, respectively.



a. CT image when feeler gauges of 330 μm are inserted.



b. CT number plot along the red line of the CT image. The horizontal line means the average matrix CT number, and the area of U-shaped region is the Integrated CT Signal which corresponds to aperture size.

Fig. 2.3-Integrated CT Signal.

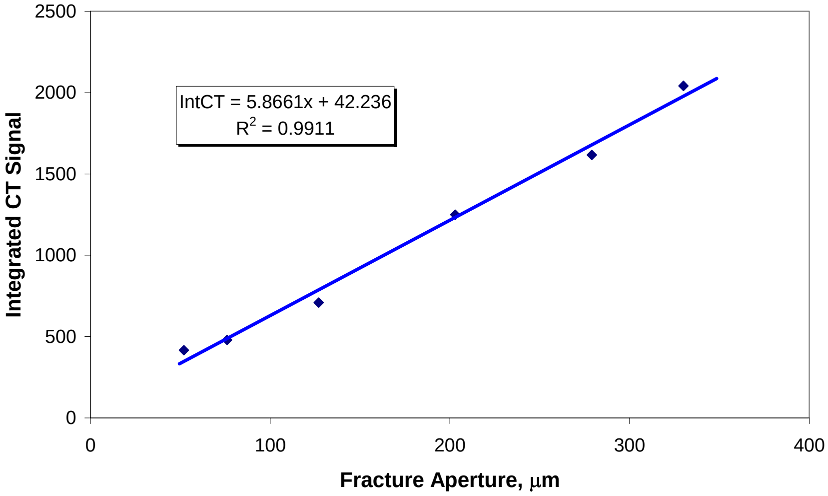


Fig. 2.4-Calibration between Integrated CT Signal and fracture aperture (Spraberry sandstone).

2.1.2 CT Scanning of Artificially Fractured Core

The scanned core was retrieved from depth of 7615.3 ft in the Spraberry Trend Formation, west Texas. From a 7-cm diameter core, a plug was obtained of which the diameter is 2.42 cm and length is 12.2 cm. An artificial tensile fracture was generated using an MTS loading machine **(Fig. 2.5)**. In order to characterize the generated fracture, the procedures of the experiment follow:

(1) CT images were taken under 1.4 MPa, 2.4 MPa, 4.1 MPa, 5.5 MPa, 6.9 MPa, 10.3 MPa, 13.8 MPa, and 17.2 MPa of confining pressure.

(2) Fracture aperture distributions were calculated, and aperture contour maps were generated under various confining pressures using the calibration curve.

Fig. 2.5-Generation of an artificial fracture using MTS loading machine. Fracture is generated perpendicular to the loading plate and along the longitudinal direction.

2.2 Fractal Analysis of Aperture

2.2.1 Self-Affine Fractal Property of Aperture

Normally, fractals can be categorized into self-similar and self-affine fractals. A self-similar fractal conserves its statistical similarities among various scales. Its scale invariant property can be expressed by the scale factor which is the same for all directions. On the other hand, a self-affine fractal conserves statistical similarities only if it is scaled differently for different directions[35]. Fractures in rock have been known to exhibit self-affine properties in several studies[7, 36].

In order to characterize a self-affine fractal, two parameters are required; fractal dimension D and amplitude parameter A. Fractal dimension represents the roughness change with scale. Amplitude parameter describes the variance at a reference scale. An equivalent parameter to fractal dimension is the Hurst exponent H ($0 < H < 1$). H and D are related by $D = E – H$, where E is the embedded Euclidean dimension.

2.2.2 Method of Calculating Self-Affine Parameters of Rock Fracture

A power law relation exists between the standard deviation of asperity height $S(w)$ and sampling window length w[37]. The $S(w)$ is calculated as the root mean square (RMS) value of asperity height residuals on a linear trend as in Eq. 2.2.

$$S(w) = RMS(w) = \frac{1}{n_w}\sum_{i=1}^{n_w}\sqrt{\frac{1}{m_i - 2}\sum_{j\in w_i}(z_j - \bar{z})^2} \quad \text{………………………………} \quad (2.2)$$

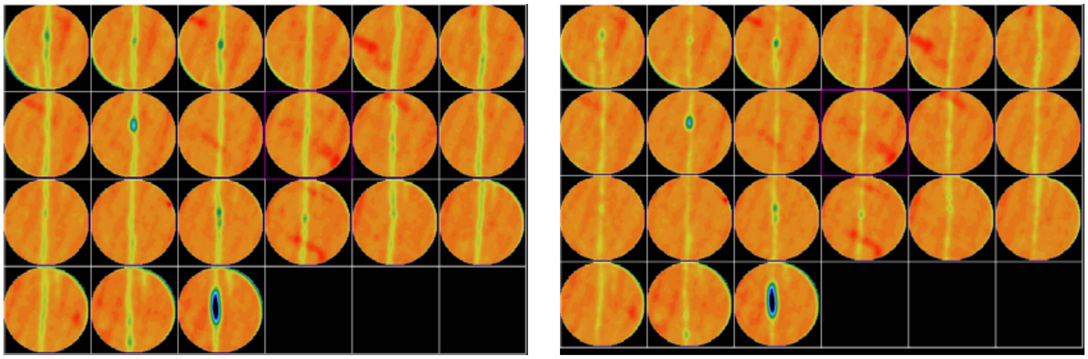
where, n_w is the total number of windows of length w, m_i is the number of points in window w_i, z_j are the residuals on the trend, and $\bar{z}$ is the mean residual in window w_i. If least square regression is used to calculate the trend, $\bar{z}$ equals to zero.

2.3 Results and Discussion

2.3.1 Scale and Pressure Effect on Artificial Fracture Aperture

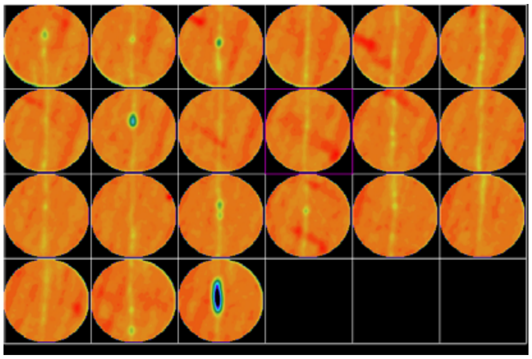
A total of 61 scans were taken for each confining pressure with a 2 mm distance between scans. **Figs. 2.6 through 2.8** show the effect of confining pressure on aperture. Fig. 2.6 shows CT images of some sections under different confining pressure. The denser region is shown with red, and less dense region is shown with yellow, green, blue, and black in decreasing order of density. The yellowish line shown in the middle of the images is the fracture. Fig. 2.6 clearly shows that low density regions, which represent fracture, are reduced by increasing confining pressure. Aperture reduction is observed in contour maps with increasing confining pressure (Fig. 2.7). The top left sections in Fig. 2.6 are the left end of the contour maps. As shown in Fig. 2.8, the modal values move to the left with increased confining pressure. By all indication, aperture distribution follows log-normal distribution as demonstrated by Muralidharan *et al.*[9].

To investigate the effect of scale and pressure on fracture aperture distribution, the sampling area of the fracture was increased from 680 mm^2 to 2952 mm^2 by 243 mm^2. A total of 10 different sampling sizes were analyzed. **Fig. 2.9** shows the results of the analysis. Fractal dimension D and amplitude parameter A tend to decrease with increased sampling area. However, beyond a certain (about 1500 mm^2) sampling area, both parameters tend to be stable. It seems that fractal dimension and amplitude parameter reach the stationary threshold when confining pressure is 17.2 MPa. Fig. 2.9 also clearly shows the effect of confining pressure. Both parameters are different under different confining pressures for each reference sampling area. Therefore, in situ confining pressure should be considered in order to properly characterize fracture aperture.



a. 1.4 MPa.

b. 6.9 MPa.



c. 17.2 MPa.

Fig. 2.6-CT images of the Spraberry core plug under various confining pressures.

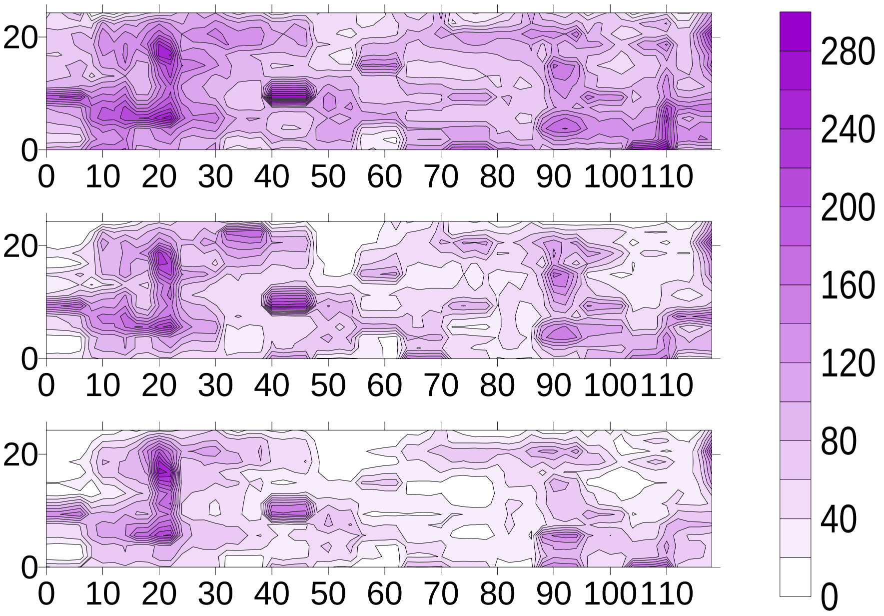


Fig. 2.7- Contour maps of the fracture aperture under confining pressure of 1.4 MPa, 6.9 MPa, and 17.2 MPa, respectively (aperture size is in μm and coordinates are in mm). Note decreasing aperture values as confining pressure increases.

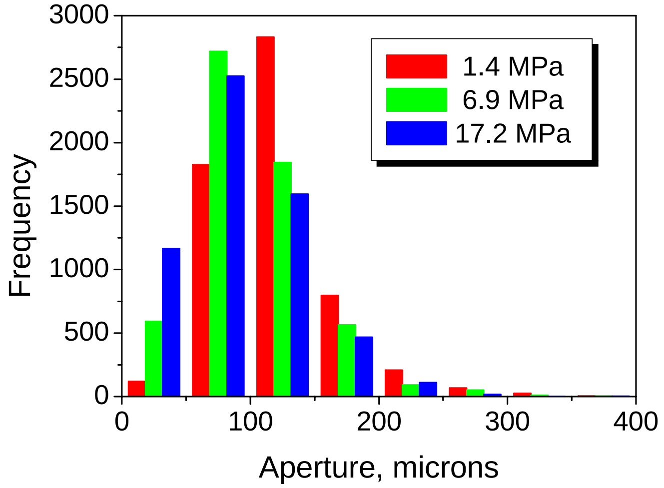
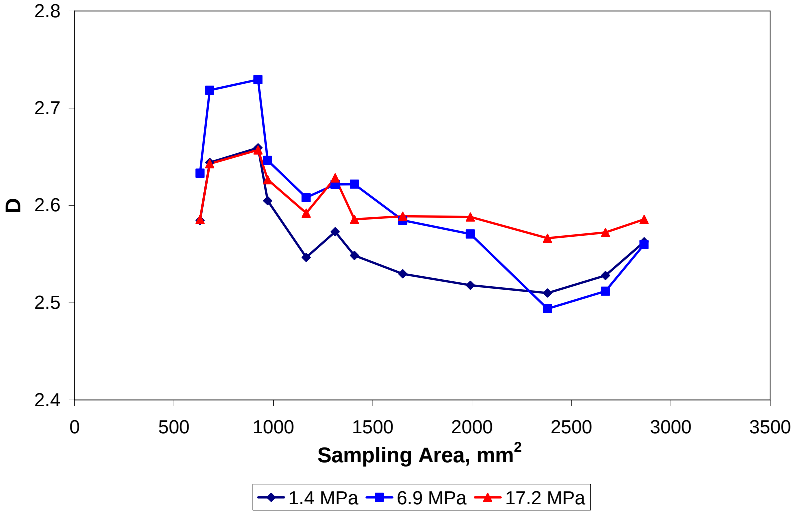
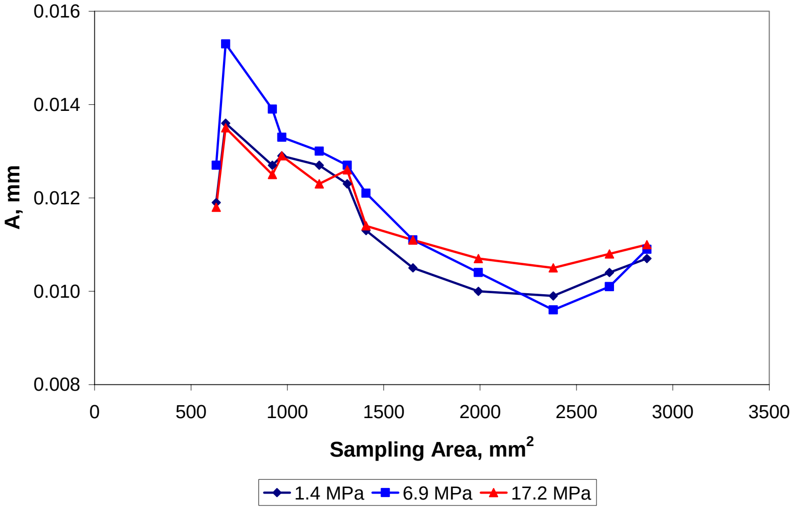


Fig. 2.8- Histograms of the fracture aperture under various confining pressures.



a. Change of fractal dimension.



b. Change of amplitude parameter.

Fig. 2.9- Change of fractal dimension, *D*, and amplitude parameter, *A*, as a function of sampling area and confining pressure.

2.3.2 Normal Fracture Stiffness

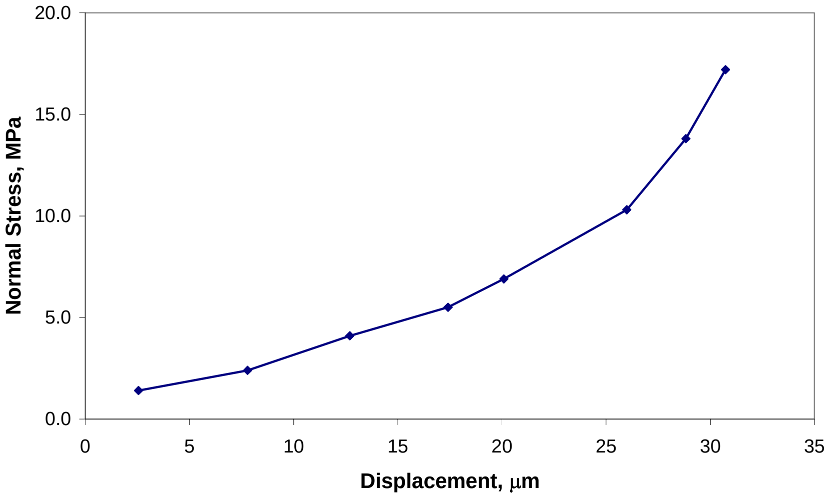
The conventional experimental method uses LVDT (linear variable differential transformers) to measure average fracture displacement. In this study, aperture contour maps from CT data were used to calculate fracture deformation. Fracture deformation was calculated by subtracting the fracture aperture of various confining pressure cases from the zero confining pressure case, then taking the arithmetic average of the subtraction. Rock deformation is neglected because the matrix portion is not big enough to deform significantly.

Fig. 2.10 shows the behavior of fracture deformation and normal fracture stiffness (K_n) as a function of normal stress. Fracture displacement increases rapidly at low normal stress and increases little at high normal stress. Normal fracture stiffness increases with increased normal stress. This behavior is the same as the behavior measured by the conventional method[38]. While the conventional method cannot measure mechanical and hydrological properties simultaneously, an X-Ray CT Scanner enables measurement of these properties at the same time without damaging the specimens. This result will be useful for fracture flow experiments. In addition, fracture compressibility (c_f), one of tricky parameters to measure, can be calculated using Eq. 2.3.

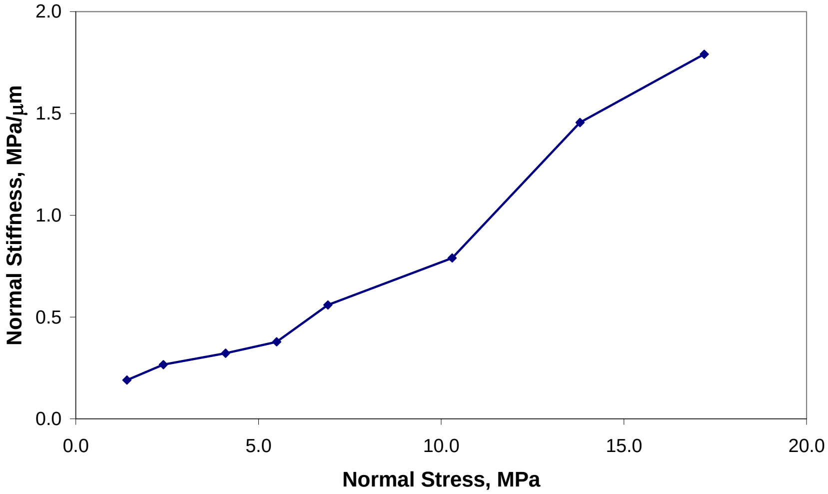
$$c_f = -\frac{1}{V_f}\frac{\partial V_f}{\partial p} = \frac{1}{Ah}\frac{A\partial h}{\partial p} = \frac{1}{h}\frac{\delta}{\partial \sigma} = \frac{1}{hK_n} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (2.3)$$

$$K_n = \frac{\sigma}{\delta}$$

where, σ is normal stress, V_f is fracture volume, p is pressure, A is area of fracture, h is height or aperture, and δ is displacement.



a. Fracture displacement.



b. Normal fracture stiffness.

Fig. 2.10-Fracture deformation and normal fracture stiffness as a function of normal stress of the artificial fracture.

CHAPTER III

ANALYSIS OF NATURAL FRACTURE APERTURE

3.1 Overview of Teapot Dome

The structure of the Teapot Dome is an elongated anticline, striking NNW-SSE, probably produced by a reverse fault whose strike is also NNW-SSE, and is located in the southwestern part of the anticline[39] **(Fig. 3.1)**. This reverse fault is dipping roughly to the NE and is considered the main fault. There are also three small faults that represent accommodation faults produced after the movement of the main fault. Most fractures generally terminate vertically at bedding planes and stylolites.

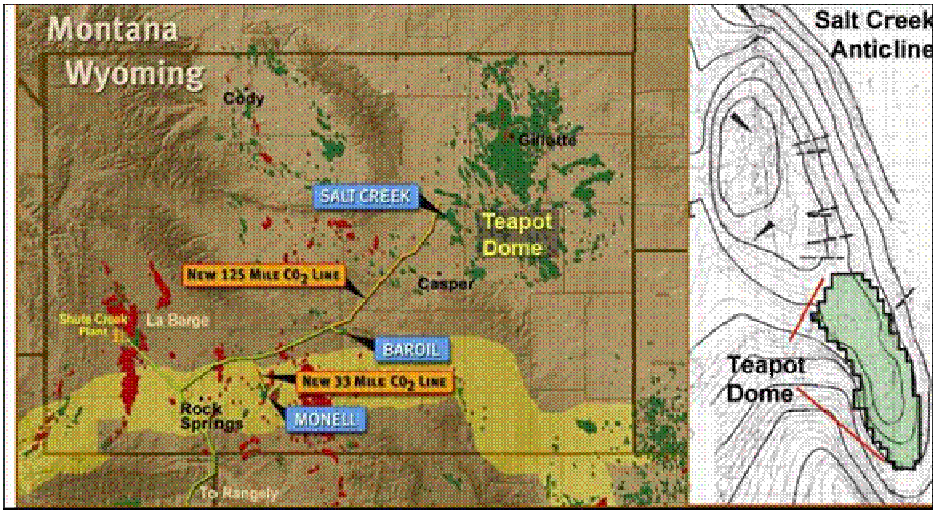
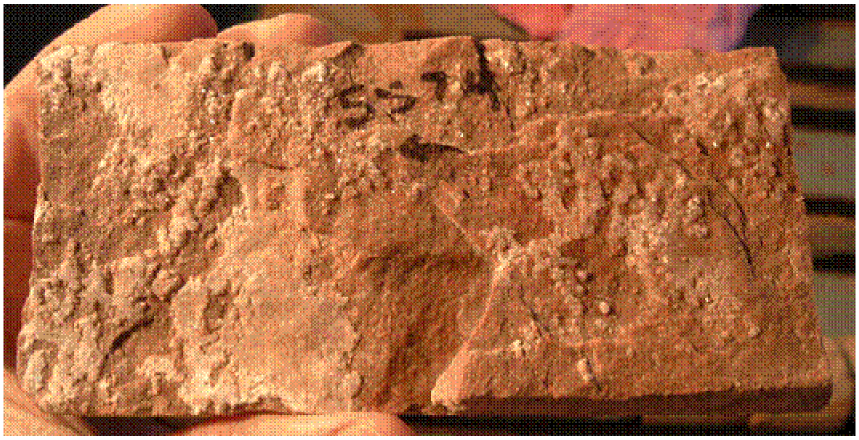


Fig. 3.1-Teapot Dome, NE Natrona Co., central Wyo., 30 mi/48 km NNE of Casper, WY. The teapot-shaped rock formation is also shown (From *Putra et al.*[39]).

The Tensleep sandstone is observed to be heavily fractured[40] **(Fig. 3.2)**. Most of the fractures are less than a millimeter in total width, and much of that width is occluded in the smaller fractures by partial mineralization of quartz and/or dolomite. Nevertheless significant porosity in the range of 10-80% remains in most fractures especially within the larger ones[40]. The larger fractures also typically split the rock and core is no longer intact across the fracture plane indicating that mineralization provides only an incomplete and weak seal between fracture faces. A zone of inclined fractures is present between 5591 ft and 5595 ft and these fractures are suggestive of a conjugate shear system similar to that seen in Tensleep outcrops immediately south of the Alcova reservoir[40]. A short of interval of white, micritic dolomite with numerous bitumen-stained natural fractures is present between 5495 and 5498 ft overlying the oil-saturated reservoir facies.



a. A natural fracture face that is partially covered with crystalline dolomite.

b. Highly fractured Tensleep sandstone.

Fig. 3.2-Cores from Tensleep formation (From Lorenz & Cooper[40]).

3.2 Experimental Procedures of Measuring Aperture Using X-Ray CT Scanner

3.2.1 Calibration

The same calibration procedures explained in Section 2.1.1 were used. **Fig. 3.3** shows the calibration results of Tensleep sandstone. The plot shows a linear relationship, as well. The correlation of aperture derived from this calibration curve for Tensleep sandstone is expressed by Eq. 3.1:

$$Fracture\ \ Aperture\,[\mu m] = \frac{IntCT - 360.99}{6.546} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots..\ldots\ldots\ldots\ldots\ldots \quad (3.1)$$

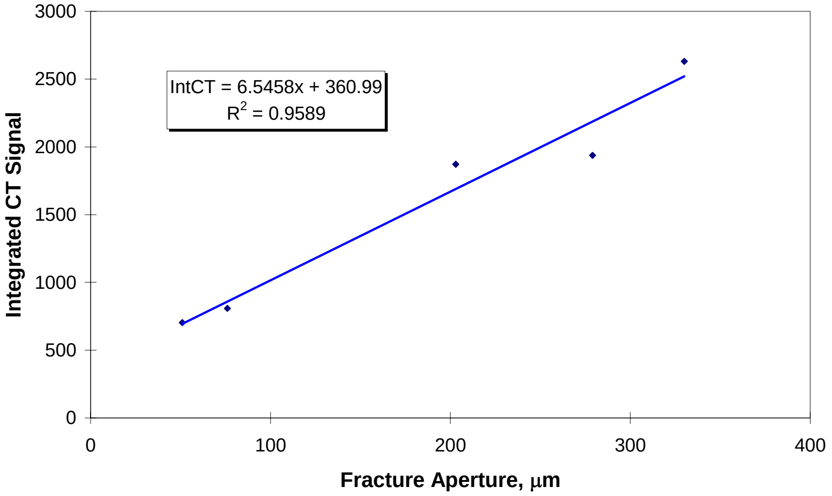


Fig. 3.3-Calibration between Integrated CT Signal and fracture aperture (Tensleep sandstone).

3.2.2 CT Scanning of Natural Fractures of Tensleep Sandstone Cores

The cores scanned were retrieved from depth between 5565 ft (Core-A) and 5566 ft (Core-B). Diameter of both cores is 6.3 cm and the length of Core-A **(Fig. 3.4)** and Core-B **(Fig. 3.5)** is 13.88 cm and 19.1 cm, respectively. CT images of the cores were taken under no overburden pressure condition.

3.3 Results and Discussion

3.3.1 Scanned Image Analysis

CT images of two cores were taken. A total 30 scans were taken for Core-A as well as 15 images contained the information of fracture. For Core-B, 17 images contained fracture among a total 50 scans were taken **(Fig. 3.6)**. **Figs. 3.7 and 3.8** show single CT image of Core-A and combination of multiple images that form a 3D image of Core-A.

Fig. 3.5 reveals that the fracture of Core-A is not extended to the outside of the core. Mineral does not precipitate inside of the fracture. The CT images also show a large void space that is seen as a cloud like shape and the open fracture looks like a board in 3D imaging (Fig. 3.8). The upper part of the CT images shows mineralized fractures that have red or orange colors as observed in Fig. 3.7. Core-B shows a set of open fractures and a set of mineralized fractures. As discussed in core analysis, Core-B fractures were precipitated with highly dense minerals like dolomite **(Figs. 3.6 and 3.9)**. Mineralized fractures are extended to the outside of the core. These fractures are parallel then join together. The open fracture is also extended to the side of the core and terminated by the mineralized fracture (Fig. 3.9). It seems that the open fracture was secondary set or caused by drilling induced fracture. **Fig. 3.10** shows a 3D image of the open fracture.

An average rock CT number is required to determine the aperture size of a fracture as explained in Chapter II. In order to get a proper value, the average rock CT number is taken based on the function of pixel position as shown in **Fig. 3.11**. This figure shows that different CT numbers present at different pixel position in the core. Once the average CT numbers were obtained the area below the average CT line were calculated. **Fig. 3.12** shows the aperture distributions of both cores. The figure shows that the open fracture of Core-B is wider and distributes more evenly than the open fracture of Core-A. We also plotted the fracture aperture contour maps of Core-A and Core-B as shown in **Figs. 3.13 and 3.14**, respectively. The results also confirm that the range of fracture aperture sizes of Core-A is much less than the size in Core-B since the open fracture in Core-B that extends to the outside of the core was probably caused by man-made induced fracture as mentioned earlier.

a. Core-A.

b. Core-B.

Fig. 3.4-Measured cores from Tensleep Formation.

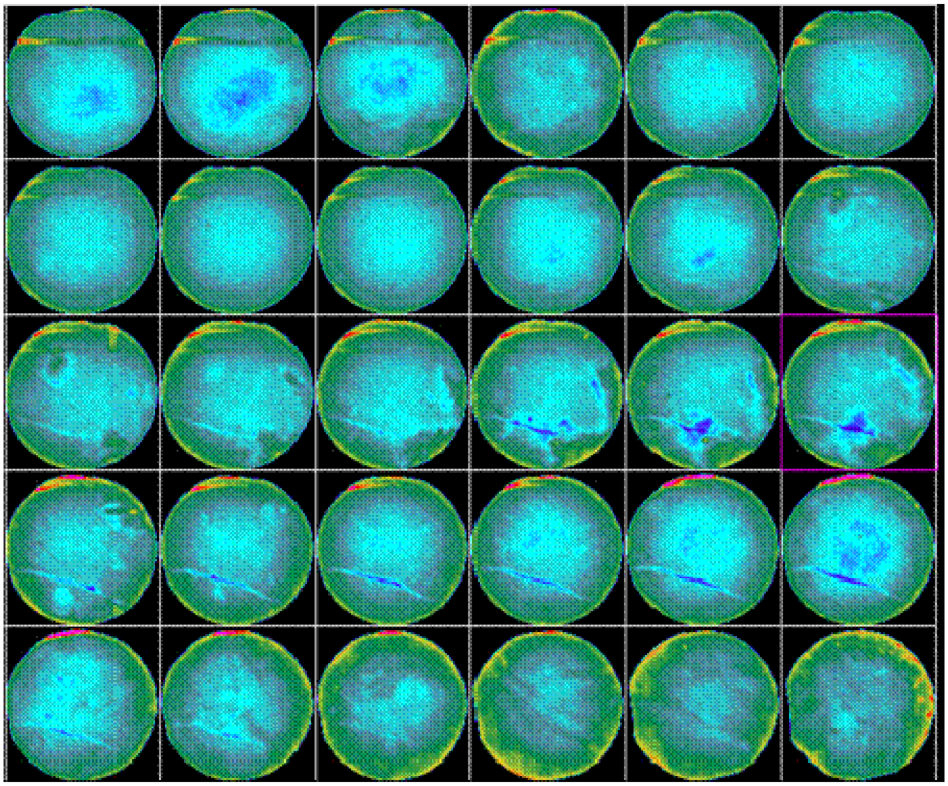


Fig. 3.5-CT images of Core-A.

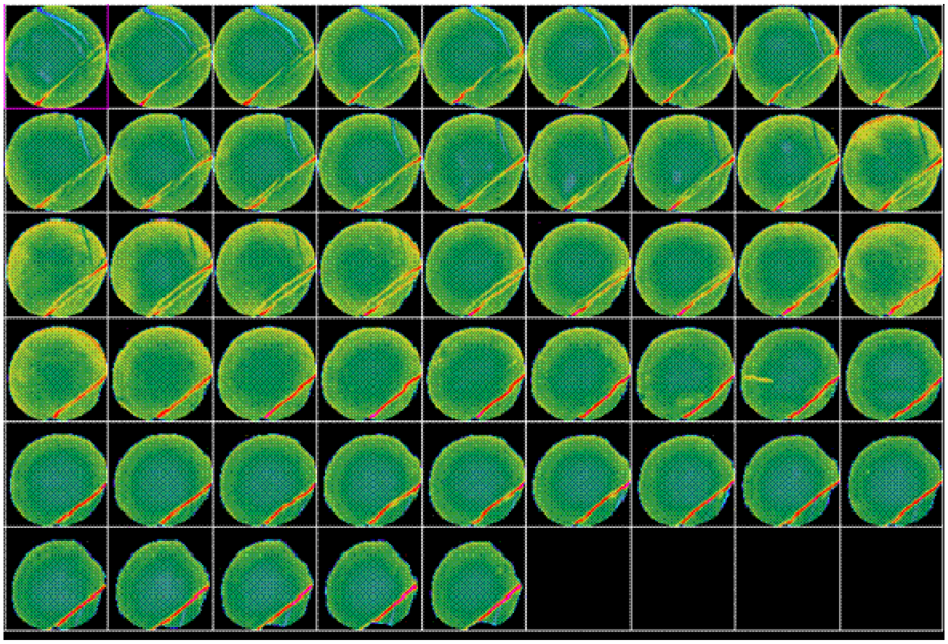


Fig. 3.6-CT images of Core-B.

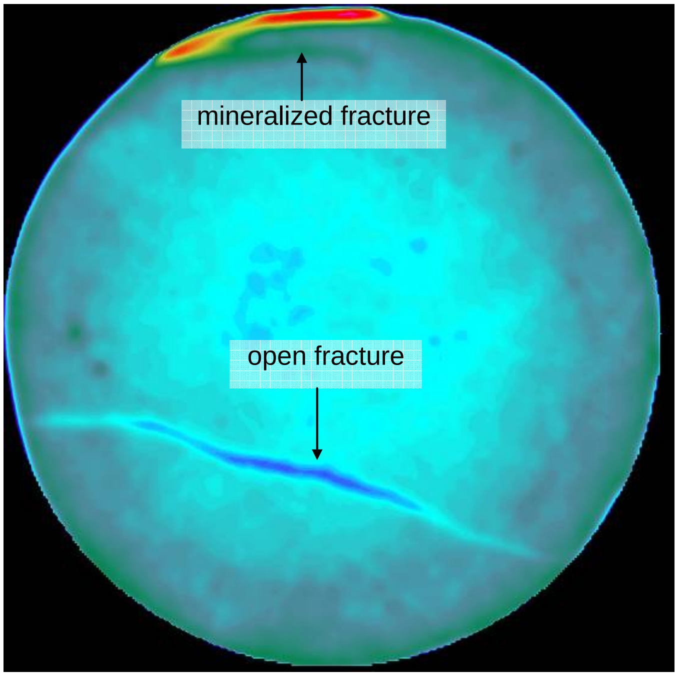


Fig. 3.7-Single CT image of Core-A.

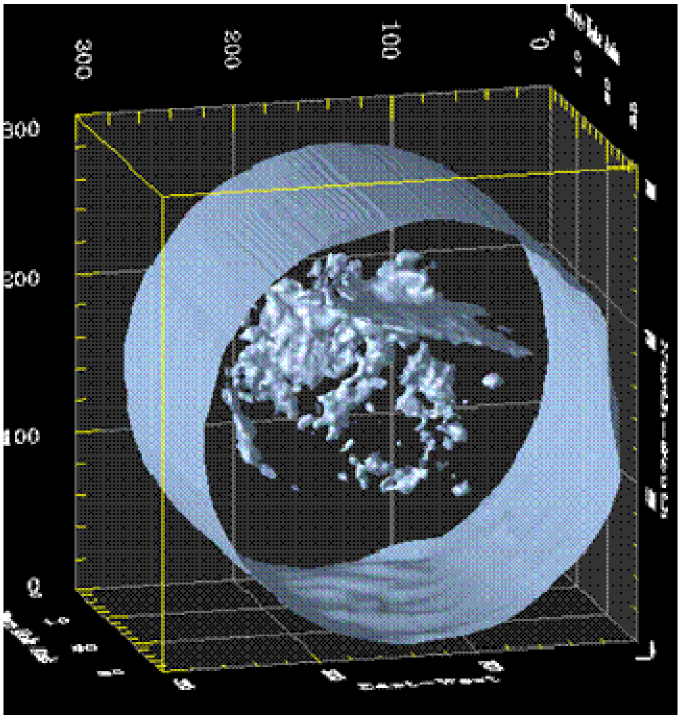


Fig. 3.8-3D image of the fracture of Core-A.

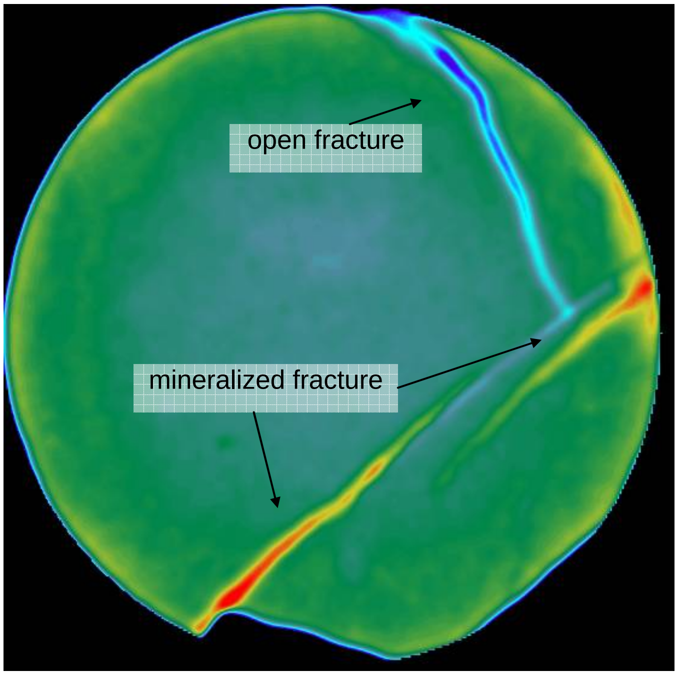


Fig. 3.9-Single CT image of Core-B.

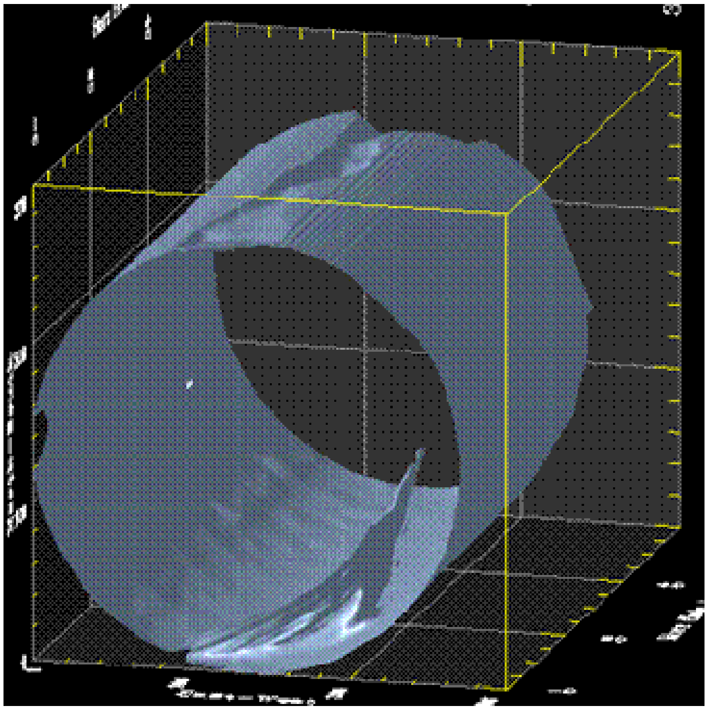
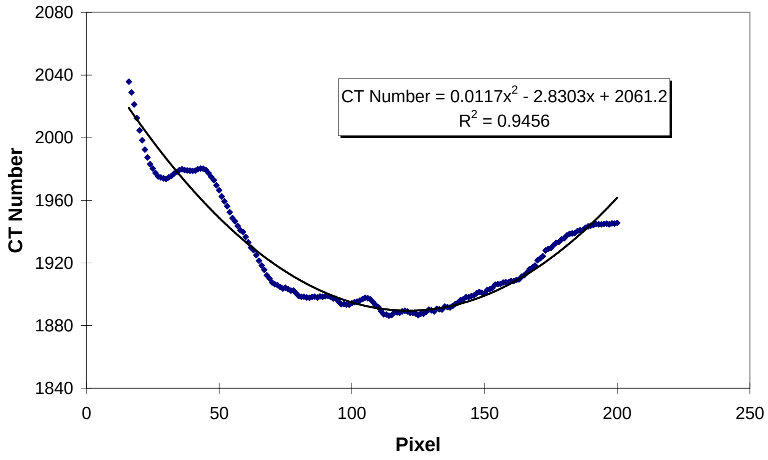
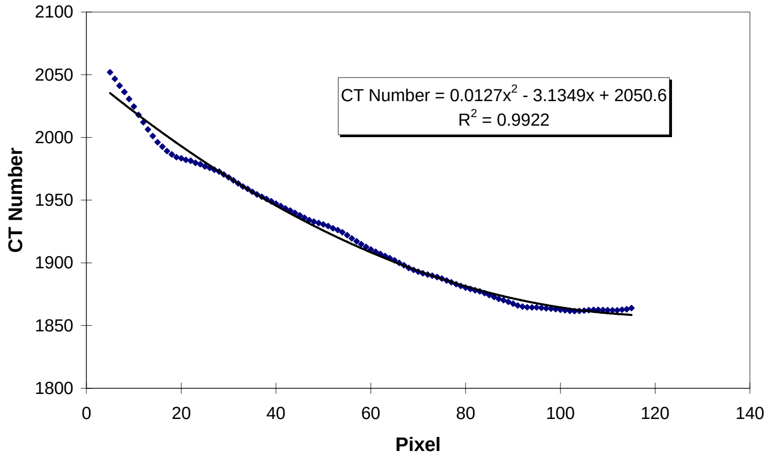


Fig. 3.10-3D image of the fracture of Core-B.



a. Core-A.



b. Core-B.

Fig. 3.11-Average CT numbers according to locations.

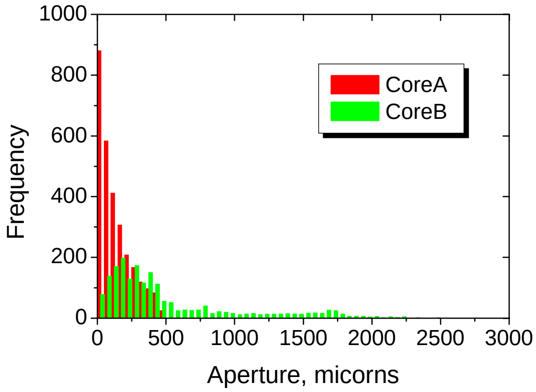


Fig. 3.12-Comparison of aperture distributions between Core-A and Core-B.

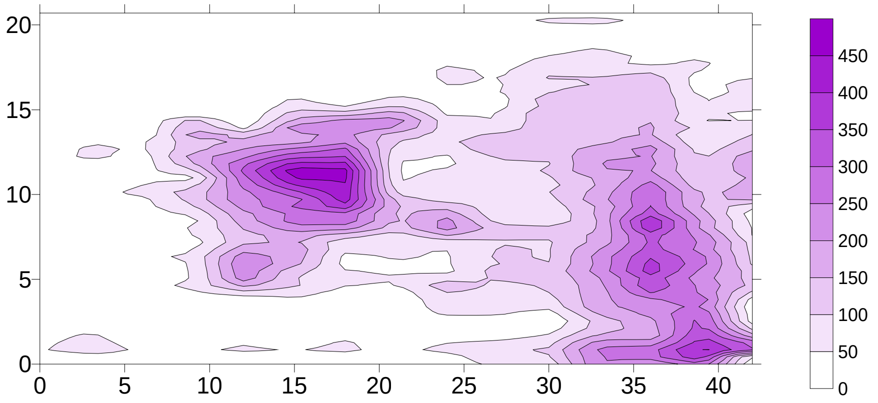


Fig. 3.13-Aperture contour map of Core-A (aperture size is in μm and coordinates are in mm).

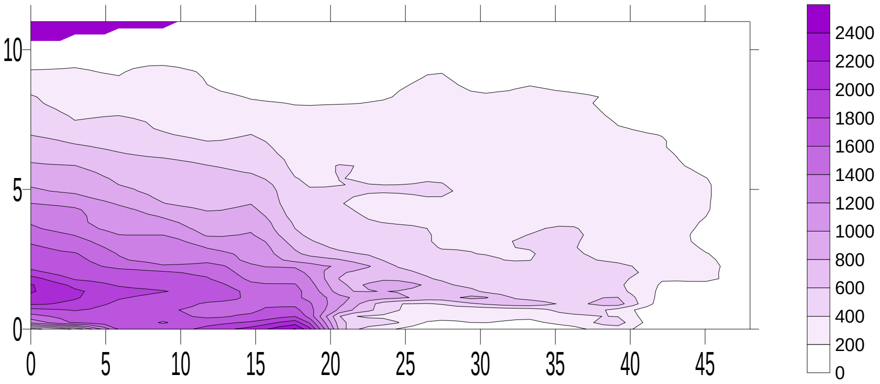


Fig. 3.14-Aperture contour map of Core-B (aperture size is in μm and coordinates are in mm).

3.3.2 Scale Effect on Natural Fracture

Using the same procedures, the effect of scale on fracture aperture distribution was examined. The sampling area of the fracture was increased from 311 mm^2 to 869 mm^2 by 186 mm^2. A total of 4 different sampling sizes were analyzed. Since the area of natural fractures is smaller than the area of the artificial fracture, the number of sampling sizes is less than that of the artificial fracture. **Fig. 3.15** shows the results of the analysis. Fractal dimension D and amplitude parameter A change with increased sampling area. While D and A of the artificial fracture tend to be stable, both parameters do not seem to be stable because of smaller area of the fractures. However, both parameters clearly show change with increasing sampling area similar to the cases of the artificial fracture. Thus, it can be said that the stationary threshold can be found, if big enough sampling area is analyzed. In addition, confining pressure may affect aperture distribution of natural fractures, by analogy.

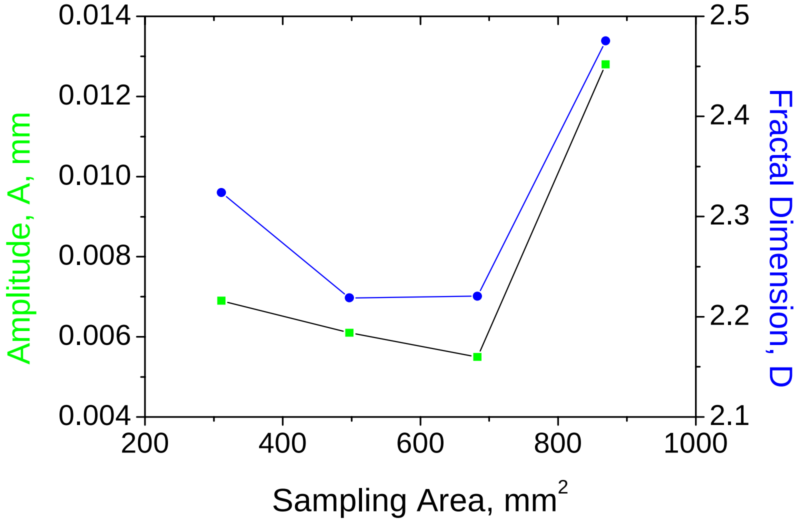


Fig. 3.15-Change of amplitude parameter and fractal dimension as a function of sampling area.

CHAPTER IV

GENERATION OF 2D FRACTAL DISCRETE FRACTURE NETWORKS

4.1 Two-Dimensional Fractal Discrete Fracture Networks (FDFN) Generation

4.1.1 Theoretical Model for FDFN

In fractal fracture network analysis, there are two approaches. One is a fractal length distribution, and the other is a fractal center distribution. For the purpose of synthetic DFN generation, the fractal length distribution model has a disadvantage. The density term of fractal length distribution depends on scale, so measured fracture density values cannot be used directly in generation with different scales. On the other hand, a fractal center distribution is easy to apply and has a constant density term regardless of scale.

In this research, a first-order model is used as a theoretical model. First-order model expressed by Eq. 4.1a (Eq. 4.1b is an integrated form of Eq. 4.1a)[41]:

$$n(l,L)dl = \alpha L^{D_c} l^{-(D_l+1)} \quad \text{……………………………………………………….} \quad (4.1a)$$

$$N(L) = \frac{\alpha}{D_l} L^{D_c} l_{\min}^{-D_l} \quad \text{……..…………………………………………………….} \quad (4.1b)$$

where, $n(l,L)dl$ is the number of fractures for which length is between l and dl in a domain of size L, D_c is a fractal dimension of fracture center distribution, D_l is a fractal dimension of the length distribution, α is the fracture density term, l_{min} is the minimum fracture length, and $N(L)$ is the number of fractures whose length is longer than l_{min} in a domain of size L. The biggest advantage of this model is that the density term, α, is a constant value regardless of the scale of measured or generated domains[26].

4.1.2 Calculation of Fractal Dimension of Fracture Center Distribution

Pair correlation function[42] is used to calculate a fractal dimension of fracture center distribution. Eq. 4.2 defines pair correlation function and shows how to calculate fractal dimension from the function[27].

$$C_2(r) = \frac{2N_p(r)}{N(N-1)} = cr^{D_C} \quad \text{……………………………………………………} \quad (4.2)$$

where, $N_p(r)$ is the number of pairs of points whose distance is less than r, N is total number of points, and c is a proportional constant.

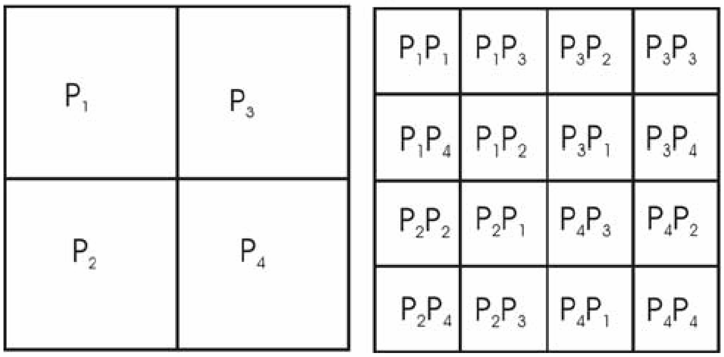
4.1.3 Fracture Center Generation

In this research, multiplicative cascade process[27, 43-45] is used to generate fracture centers following a first-order model. Multiplicative cascade process is an iterative method. This process divides a domain into subdomains until the number of subdomains reaches the desired number **(Fig. 4.1)**. The parent domain is divided into subdomains and a probability, P_i is randomly assigned to each subdomain (Fig. 4.1a). At the next step, subdomains become parent domains and are divided into smaller subdomains. Then, P_i is randomly assigned and multiplied with the probability of a parent domain (Fig. 4.1b). Such probabilities are calculated by Eq. 4.3[27, 45].

$$\sum_{i=1}^{n} \frac{P_i^q}{(1/sr)^{(q-1)D_q}} = 1 \quad \text{………………………………………………………………} \quad (4.3)$$

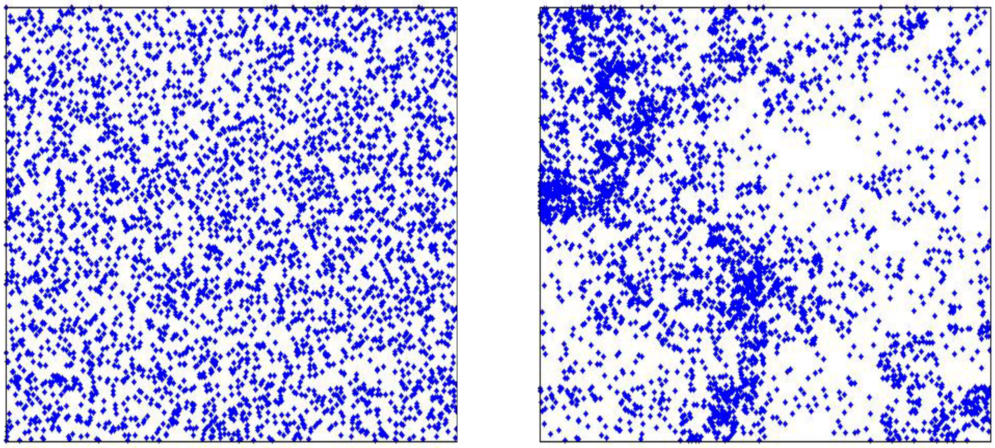
where, P_i is a probability, sr is the scale ratio (ratio between the side length of a parent domain and subdomain), and D_q is a multi-fractal dimension (in this research 2nd dimension is used, so q is fixed at 2). Fig. 4.1 shows the case when the scale ratio, sr is 2. When the scale ratio is 3, all procedures are the same but parent domains are divided into 9 subdomains.

In many researches, a Poisson process is adopted in DFN generation[31, 32]. Poisson process assumes that fracture centers distribute randomly across a domain. The main advantage of a multiplicative cascade process is that it can mimic the nature of fracture clustering which a Poisson process cannot. In the nature, fractures tend to form groups and clusters[15, 46], so this advantage leads to more realistic DFN generations. **Fig. 4.2** compares the fracture center distributions following a Poisson process and multiplicative cascade process. While fracture centers generated using a Poisson process distribute uniformly across the domain, fracture centers generated using a multiplicative cascade process mimic the clustered nature well. Therefore, it can be said that a multiplicative cascade process generates more realistic FDFN than does a Poisson process.



a. First iteration b. Second iteration

Fig. 4.1-The example of multiplicative process with the scale ratio, *sr*=2 (From Darcel *et al.*[27]).



a. Poisson process b. Multiplicative cascade process

Fig. 4.2-Fracture center distributions generated by a Poisson process (a) and a multiplicative cascade process (b).

4.1.4 Fracture Orientation

Fracture orientation in 2D is expressed using strike only, while dip and dip direction are used in 3D representation. In this research, fracture orientation is assumed to follow Fisher distribution[31]. Fisher density function is expressed by Eq. 4.4[31].

$$f(\theta) = \frac{K \sin\theta \, e^{K\cos\theta}}{e^{K} - e^{-K}} \; (0 < \theta < \pi/2) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4.4)$$

where, θ is angular deviation from the mean, and K is Fisher's constant which is calculated by analyzing field data such as outcrop maps and shows the extent of deviation from the mean. To generate fracture orientations, fracture angular deviation, $R^{i}_{F,K}$ from the mean is calculated by Eq. 4.5[31] in degrees and is added to the mean.

$$R^{i}_{F,K} = \cos^{-1}\left(\frac{\ln(1 - R^{i}_{G,1})}{K} + 1 \right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4.5)$$

where, $R^{i}_{G,1}$ is a Gaussian random number between 0 and 1.

4.1.5 Fractal Aperture Generation

Fracture aperture in many DFN generations is a single value for one fracture. This approach assumes fractures are parallel plates. However, this approach is far from real fracture behaviors[9] as was previously described. Single aperture assignment also assumes that the mean of aperture follows certain distribution functions, such as log normal distribution[31, 32] or fracture aperture is proportional to its length or diameter[33]. Both practices are easy and fast in DFN generation, but do not produce appropriate fracture porosity estimation. Fracture porosity generated by the former method is not consistent. When a large value is assigned to a large fracture, the fracture porosity increases and vice versa. The latter overestimates fracture porosity because the "maximum" aperture value is proportional to its length[47] for tensile fractures only. Therefore, aperture-length relation overestimates fracture porosity also, and is not appropriate other types of fractures such as shear fractures.

The most common mathematical model for self-affine fractals is the fractional Brownian motion (fBm) (Voss, 1988). The fBm, $G_H(t)$ is expressed by Eq. 4.6[48-50].

$$\left\langle \left| G_H(X+h) - G_H(h) \right|^2 \right\rangle \propto |h|^{2H}, (0 < H < 1) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4.6)$$

where, H is the Hurst exponent, h is lag or height in the case of fracture aperture generation, and the brackets, and the brackets < > mean ensemble averages over many samples of $G_H(h)$.

There are many algorithms for fBm generations such as Fast Fourier Transform (FFT) filtering, random midpoint displacement (RMD), and successive random addition (SRA). Among those algorithms, a corrected SRA algorithm is used in this research because it is relatively simple, efficient in computation, and does not have a problem of artifacts like RMD does[48, 51]. The computer codes used in this research are based on the algorithm in Saupe's work[52] with modifications done by Liu *et al.*[51].

According to the central limit theorem, a constant multiplied by a Gaussian random variable with zero mean and variance, σ^2 is still Gaussian. Using this property, the relation of the standard deviation of the left term of Eq. 4.6 and lag can be expressed by Eq. 4.7, which is used in SRA algorithm[49]. The detailed procedures of SRA algorithm[48, 52] are shown in **Fig. 4.3**. The first step is generating random numbers from $R_G(0, \sigma_0^2/2)$ at the end points (Step 1 in Fig. 4.3), where $R_G\ (0, \sigma_0^2/2)$ represents a Gaussian random number whose mean is zero and variance is $\sigma_0^2/2$. The next step is linearly interpolating the midpoint (Step 2 in Fig. 4.3) using the end points generated in Step 1. Then, add the random numbers, $R_G\ (0, \sigma_1^2/2)$ to all points. The new variance, $\sigma_1^2/2$ is calculated by Eq. 4.8. By linearly interpolating, new numbers are generated between existing numbers (Step 3 in Fig. 4.3). The random numbers, $R_G\ (0, \sigma_2^2/2)$, are added to all points; the procedure is repeated until the desired number of iterations, n, is reached (Step 4 in Fig. 4.3). The final step is adding random numbers, $R_G\ (0, \sigma_j^2/2)$, to all points. Random number addition is done until σ_j^2/ σ_0^2 is negligible. At the final step, linear interpolation is not performed. The purpose of the final step is to obtain the proper variance structure at the n^{th} step[51]. This modeling method has been used for single fracture aperture generation. To my knowledge, no one has tried to combine realistic fracture aperture models with DFN to estimate fracture porosity. **Fig. 4.4** shows some examples of generated fracture apertures. Each fracture in FDFN has its own fracture aperture profile rather than having a single aperture value.

$$\sigma^2(h) = \sigma^2(1)h^{2H} \quad \ldots\ldots\ldots\ldots \quad (4.7)$$

$$\sigma_j^2 = \frac{\sigma_{j-1}^2}{2^{2H}} = \frac{\sigma_0^2}{\left(2^{2H}\right)^j}\left(1 - \frac{2^{2H}}{4}\right) \ldots\ldots\ldots\ldots \quad (4.8)$$

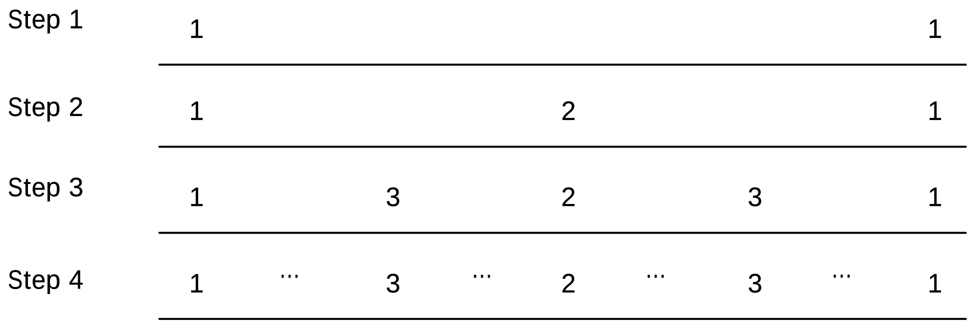


Fig. 4.3-Procedures of SRA algorithm (After Liu *et al.*[51]).

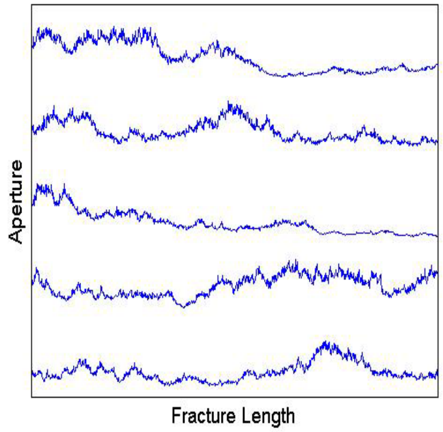


Fig. 4.4-Examples of fractal fracture aperture profiles generated by the Corrected SRA. To clearly show roughness, the figure is exaggerated vertically.

4.1.6 General Features of the 2D FDFN Generation Codes

Based on the mathematical models explained above, 2D FDFN generation codes have been developed using MATLAB®. **Fig. 4.5** is a brief flow chart of the codes (refer Appendix B for the 2D source codes). The codes require two types of input parameters. One is for the geometry of fracture networks and the other is for aperture generation. The types of input data for the fracture geometry are fractal dimensions of fracture center distribution, D_c, and fractal dimensions of length distribution, D_l, fracture density term, α, minimum fracture length, l_{min}, fracture mean orientation, and its Fisher's constant, K. Input parameters for aperture generation are Hurst exponent, H, and amplitude, A. The codes are designed to generate up to three different fracture sets. The codes also have a scanline subroutine which calculates fracture spacing along a scanline on FDFN **(Fig. 4.6)**. The scanline is drawn perpendicular to the mean orientation. There are two options of estimating fracture porosity; one is Monte Carlo simulation and the other is fractal porosity calculation. When a domain size is known, Monte Carlo simulation will provide a possible fracture porosity range with probability **(Fig. 4.7)**. Fractal dimension of fracture porosity is the option used when a domain size is uncertain **(Fig. 4.8)**. Detailed procedures are explained in Appendix B.

4.2 Validation of 2D FDFN Generation Codes

4.2.1 Self vs. Self Validation

Validation was done in two ways. The first was comparing the input data and fractal characteristics of generated FDFN to check whether the developed codes work properly. The other one was comparing the results of the generations with theoretical models which are not used in generation algorithms. Fractal dimensions of fracture center and length distributions of FDFN were calculated using Eq. 4.2 and Eq. 4.1, respectively. **Table 4.1** lists the input values and fractal characteristics of FDFN. **Fig. 4.9 through 4.12** show the generated FDFN and the fractal analysis results of case 1 and 2. Fractal dimensions of center distribution, D_c and fractal dimensions of length distribution, D_l are almost identical. Density term, α are a bit different from input but such differences are acceptable. Thus it can be said that the developed codes work properly.

Fig. 4.13 shows a rose diagram of generated 2D FDFN as an example and **Table 4.2** lists all input orientations and Fisher's constant and the results of the analysis. Fisher's constant is calculated by Eq. 4.9[31].

$$K = \frac{M-2}{M-|r_n|} \qquad \text{……………………………………………………………………} \qquad (4.9)$$

where, M is the number of fractures, and r_n is the resultant vector of the normal vectors of fractures. The results of analysis show that both parameters, strike and Fisher's constant are almost identical to the input. Thus, fracture orientations are properly generated.

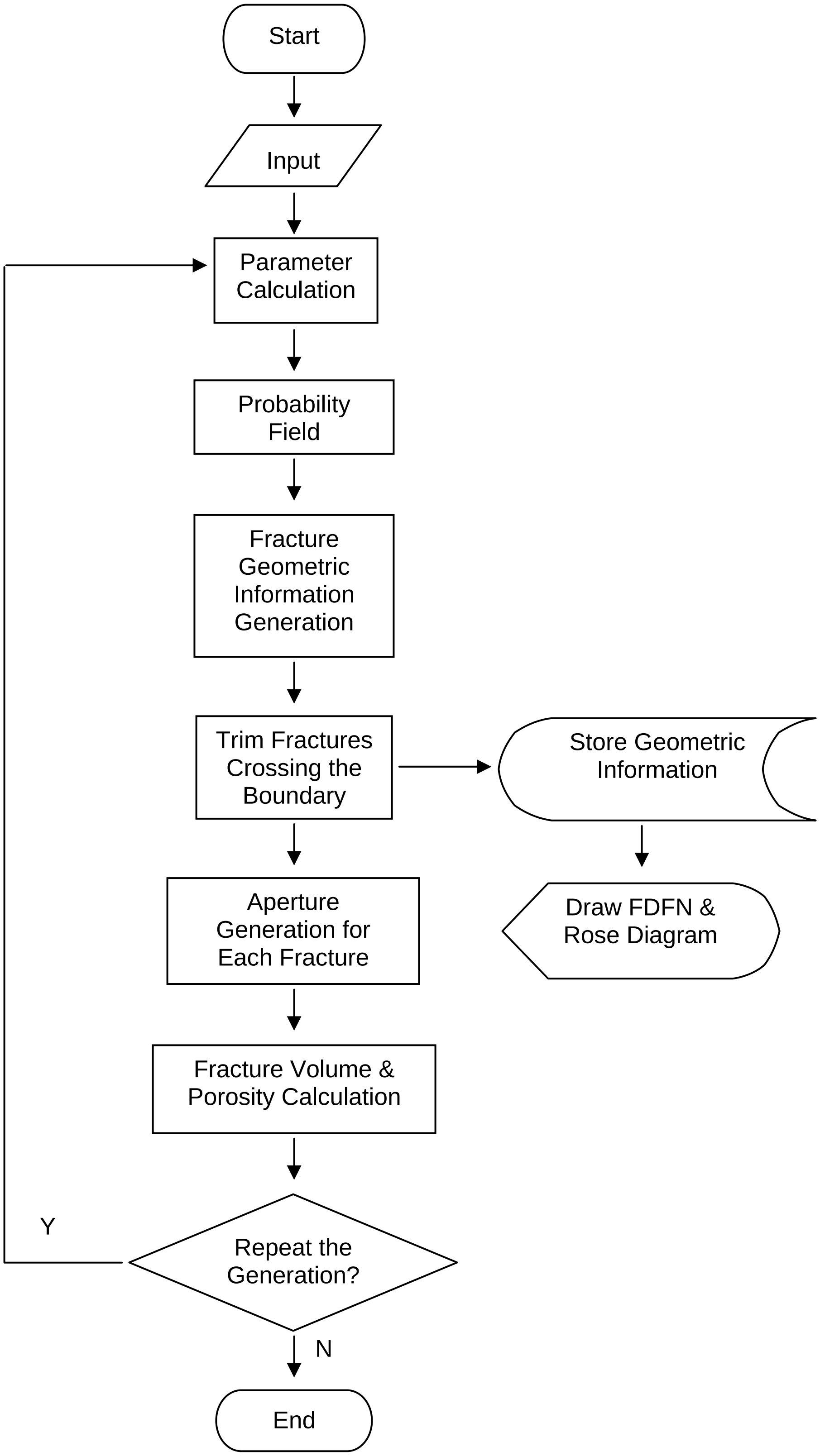
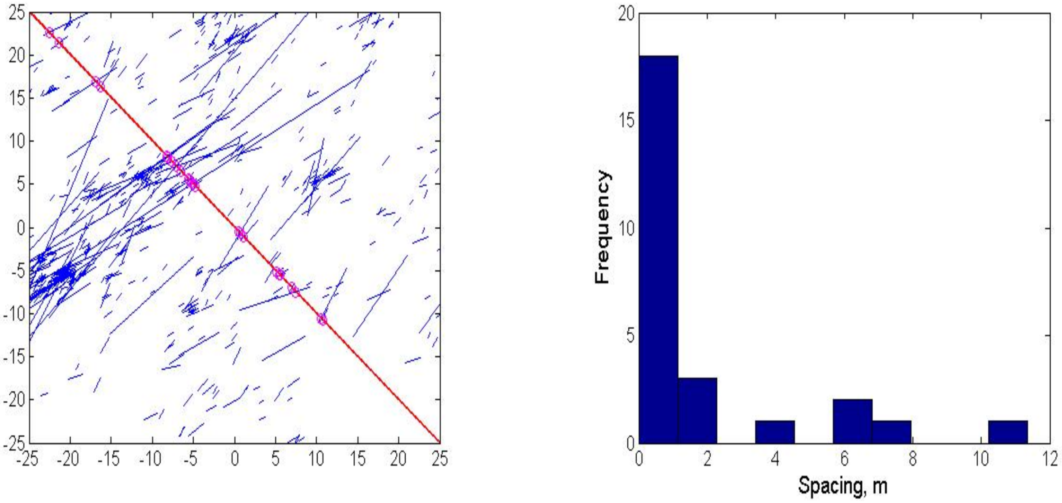


Fig. 4.5-Flowchart of 2D FDFN generation codes.



a. Sample 2D FDFN and scanline b. Fracture spacing along the scanline

Fig. 4.6-Generated FDFN's spacing characteristic, using scanline option.

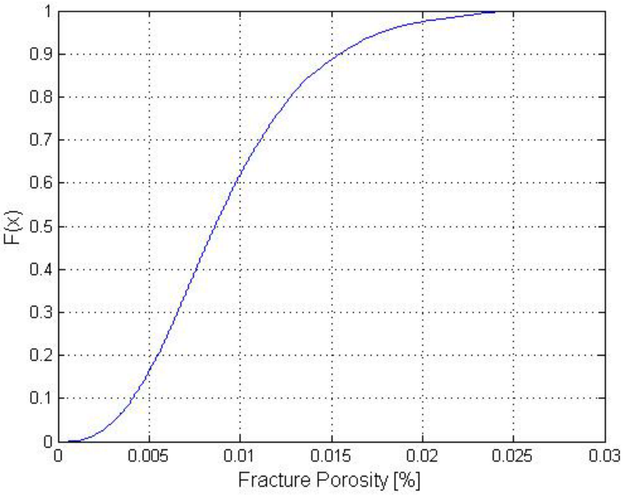


Fig. 4.7-Cumulative probability chart of fracture porosity produced by running Monte Carlo simulation option; provides fracture porosity with probability.

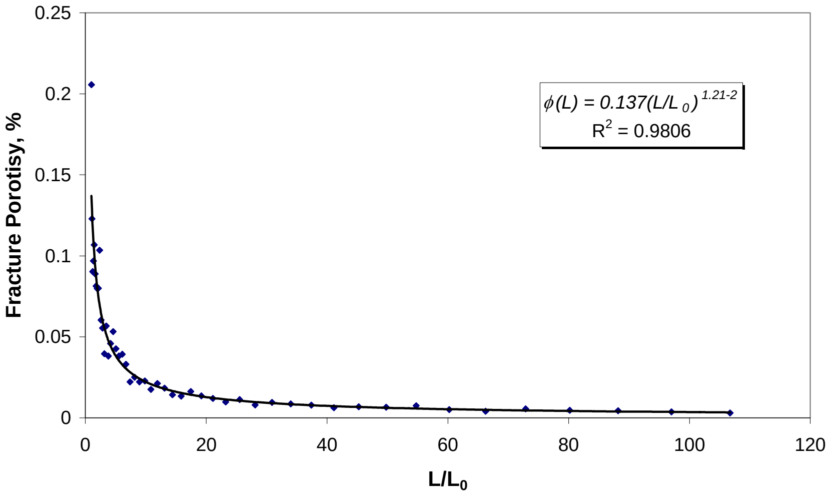


Fig. 4.8-Fractal dimension of fracture porosity: used when domain sizes are uncertain.

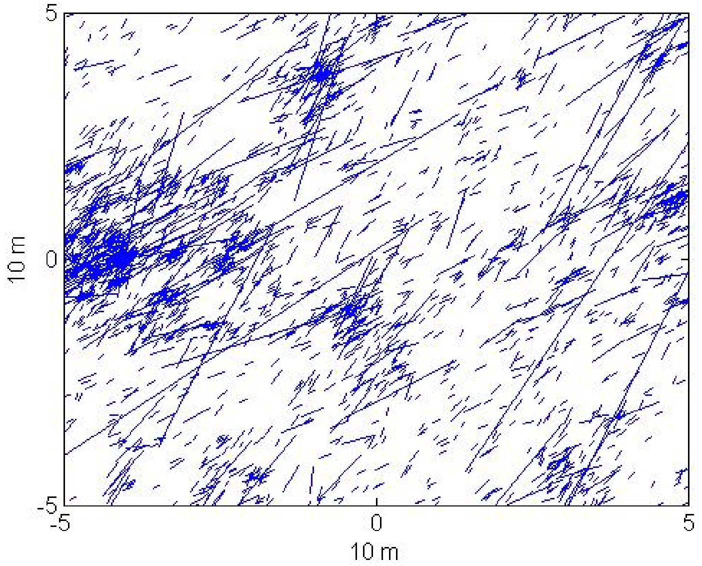


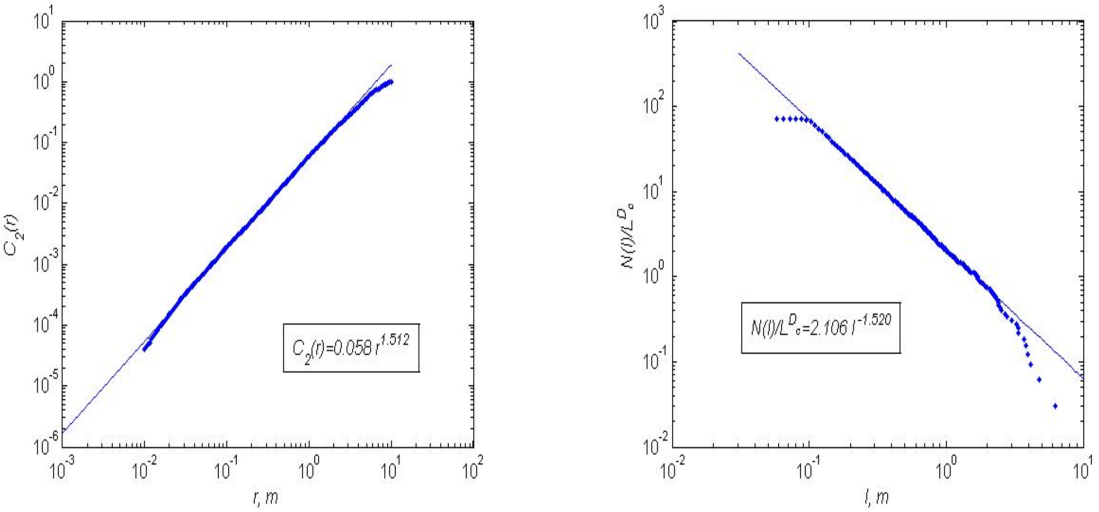
Fig. 4.9-2D FDFN of case 1. The generation dimension is 10 m by 10 m; a total of 2326 fractures are generated.

4.2.2 Validation Using Theoretical Models

Bour and Davy[53] showed that a power law relation exists between the center-to-center fracture distance and fracture lengths using San Andreas Fault data. The power law was also proved by Bour *et al.*[26] using fracture data of the Hornelen basin, Norway. Eq. 4.10[26] is the mathematical expression of the power law relation and it is used to verify the geometry of FDFN.

$$d(l) \propto l^{x},\ (x = D_l / D_c) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \tag{4.10}$$

where, *d(l)* is the distance between a fracture and its nearest neighbor having a length larger than *l*, and *x* is an exponent.



a. Fracture center distribution

b. Fracture length distribution

Fig. 4.10-Fractal analysis of 2D FDFN shown in Fig. 4.9.

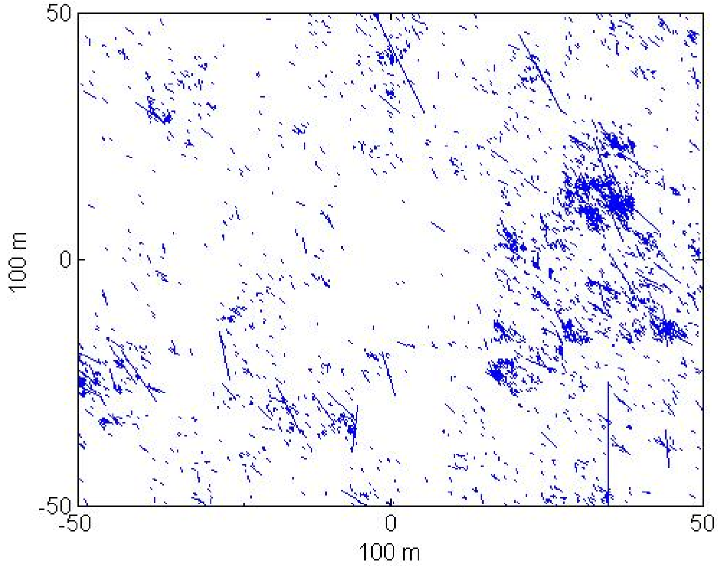
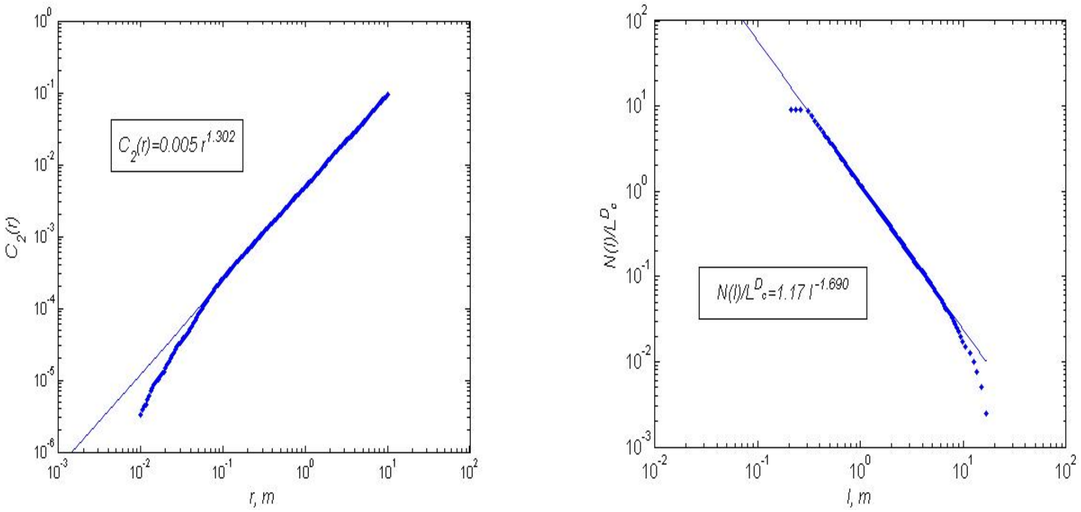


Fig. 4.11-2D FDFN of case 2. The generation dimension is 100 m by 100 m; a total of 4310 fractures are generated.



a. Fracture center distribution

b. Fracture length distribution

Fig. 4.12-Fractal analysis of 2D FDFN shown in Fig. 4.11.

Table 4.1 Comparison of Input Data and Generation Results of Fractal Dimensions

		D_c	D_l	α
Case 1	Input	1.6	1.5	3.5
	Results	1.512	1.520	3.203
Case 2	Input	1.3	1.7	2.5
	Results	1.302	1.690	1.983

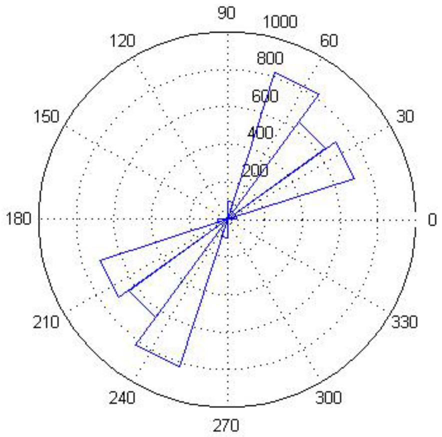
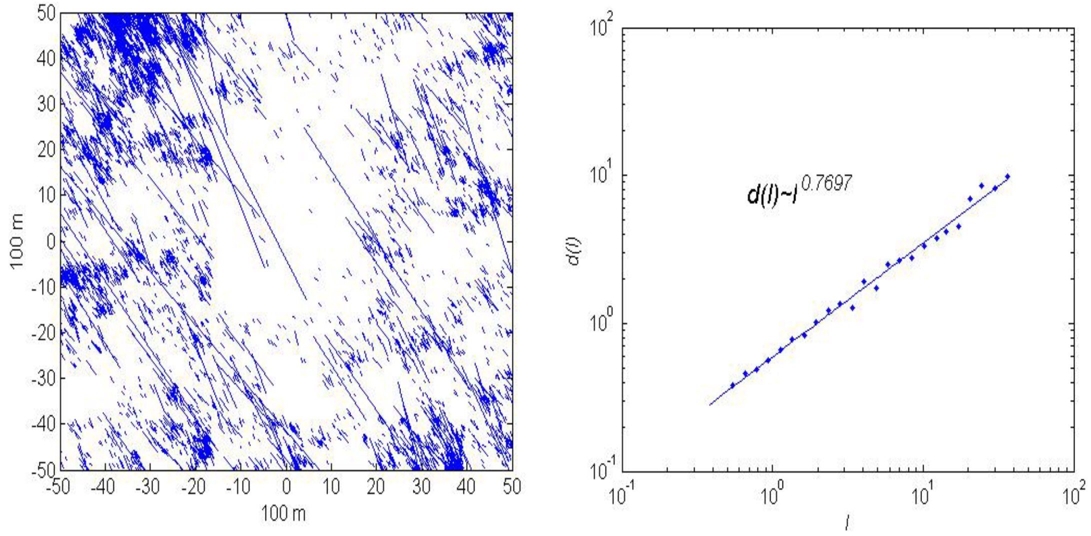


Fig. 4.13-Rose diagram of 2D FDFN of case 1, Fig. 4.9, whose mean strike is N43^{o}E.

Table 4.2 Comparison of Input Data and Generation Results of Fracture Orientation

		Strike	K
Case 1	Input	N43 ^{o}E	35
	Results	N43.31 ^{o}E	36.108
Case 2	Input	N33 ^{o}W	5
	Results	N33.23 ^{o}W	5.062



a. FDFN b. Fracture distance

Fig. 4.14-Generated 2D FDFN and its fractal analysis result of fracture distance (dimensions are in meters).

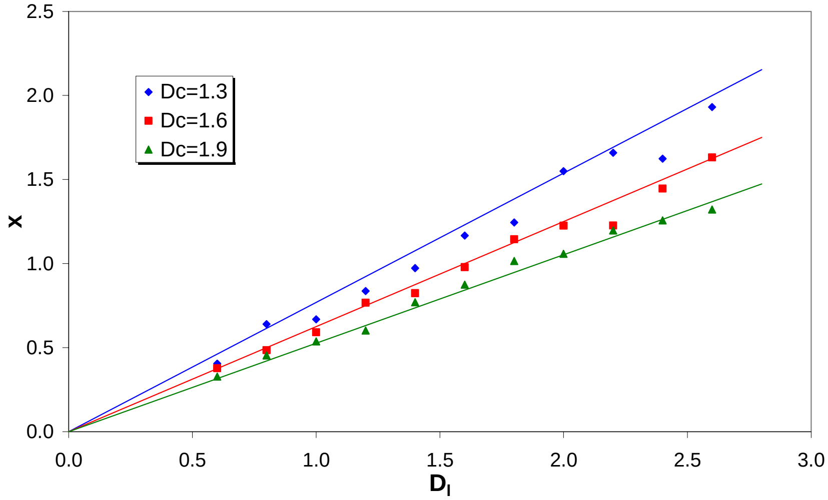


Fig. 4.15-Comparison between theoretical values and the fractal analysis results of 33 FDFN.

A total of 33 2D FDFN were analyzed to check whether generated FDFN matched well with the theoretical model. **Fig. 4.14** shows one of the 33 tested FDFN and the results of fractal analysis. The generated FDFN is 100 m by 100 m. The average distance between a fracture and its nearest neighbor having a length, l and $l+dl$ was calculated. Input of D_c and D_l are 1.9 and 1.4 respectively. The exponent, x of the generated FDFN is 0.7697 (Fig. 4.14b) and it is very close to the theoretical value, 0.7368. This case demonstrates that synthetic fracture networks mimic the characteristics of real fracture networks. **Fig. 4.15** shows the fractal analysis results of all 33 FDFN. The solid lines represent the theoretical values, and the points are the values of generated FDFN. The results of generated FDFN analysis are distributed along theoretical lines. The maximum difference between the theoretic value and generated value is less than 14%. Therefore, it can be concluded that the 2D FDFN generation code produce reasonable fracture network geometry.

The mathematical model (Eq. 4.11) suggested by Chang and Yortsos[54, 55] is used to validate the fracture porosity estimated by the codes. Eq. 4.11 shows that fracture porosity is a scale dependent value rather than a constant, and has fractal characteristics.

$$\phi_f(L) = \phi_{f0}\left(\frac{L}{L_0}\right)^{D_{fp}-E} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4.11)$$

where, L is a side length of a region, L_0 is a side length of a region where fracture porosity is ϕ_{f0}, D_{fp} is a fractal dimension of fracture porosity, and E is an embedded Euclidean dimension. In validation, 6 different cases were tested and input values of all tested cases are listed in **Table 4.3**. The side length of a generation domain started at 10 m by 10 m. Ten realizations were performed for one domain size and fracture porosity values were averaged. At the next step, the side length increased by 10% and the same procedures were repeated. These procedures were repeated for 50 different side lengths. R^2 values, one of the parameters showing the fitness of the data are good enough to tell that the trends of fracture porosity with increasing domain size are well matched with the theoretical model **(Fig. 4.16)**. In addition, Fig. 4.16 shows fracture porosity trends with different input are well matched with the mathematical model of Chang and Yortsos[54, 55]. With increasing fractal dimension of fracture center distribution, D_c, fractal dimension of fracture porosity, D_{fp} tends to decrease. On the other hand, D_{fp} decreases as D_l increases. Increasing D_c causes the increase of the number of fractures. It means that fracture porosity with high D_c is larger than the value with low D_c. However, high D_l causes the generation of short fractures and reduces number of fractures so that fracture porosity decreases.

Table 4.3 Input Data and R^2 Values of the 2D Cases

	D_c	D_l	D_{fp}	R^2
Case 1	1.3	1.4	0.8826	0.9937
Case 2	1.3	1.8	0.4621	0.9958
Case 3	1.6	1.4	1.2100	0.9806
Case 4	1.6	1.8	0.8144	0.9919
Case 5	1.9	1.4	1.4925	0.9352
Case 6	1.9	1.8	1.1016	0.9715

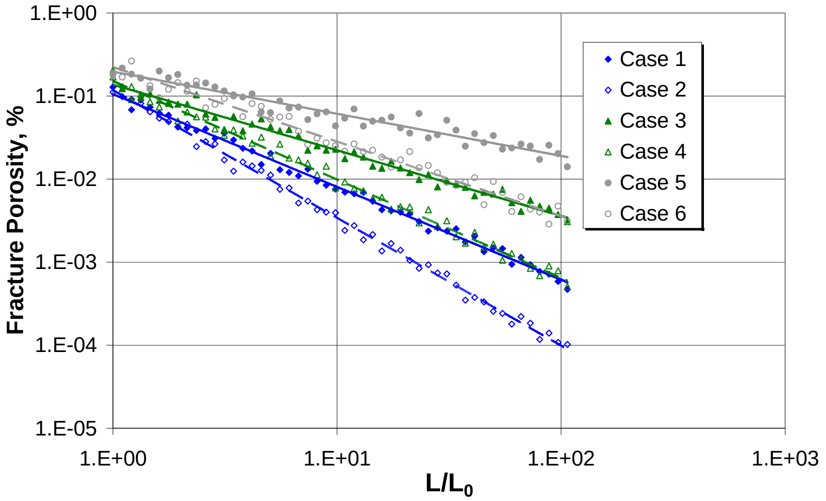


Fig. 4.16-Fractal behavior of fracture porosity. Bold lines denote the trend lines of cases 1, 3, 5, and dashed lines denote the trend lines of cases 2, 4, 6. Table 4.3 shows the input and R^2 values.

CHAPTER V

GENERATION OF 3D FRACTAL DISCRETE FRACTURE NETWORKS

5.1 Three-Dimensional FDFN Generation

5.1.1 Theories and Algorithms Used in 3D FDFN Generation Codes

Basically, the same theories and algorithms are used in 3D FDFN generation with 2D FDFN generation codes. The only difference is that Euclidean dimension increases from two to three which requires some modifications. Eq. 5.1 is the 3D version of Eq. 4.1. Fractal dimensions of Eq. 4.1 are simply added by 1 with increasing Euclidean dimension from two to three, but density term is required to be modified. In 2D, the density term is calculated using Eq. 4.1 and 4.2. However, it is impossible to use those equations directly in 3D cases because there is no mean to measure fracture diameters in 3D. To solve this problem, the relation between 2D and 3D density terms is derived (Eq. 5.2) using Eq. 4.1 and Piggott's equation[28]. The detailed derivation procedures are shown in Appendix A.

$$n(l,L)dl = \alpha_{3D} L^{D_{c3D}} l^{-(D_{l3D}+1)} \quad \text{..…………..……………………….………………...} \quad (5.1)$$

$$\alpha_{3D} = \alpha\sqrt{\pi}\,\frac{\Gamma\left(\dfrac{3+D_l}{2}\right)}{\Gamma\left(\dfrac{2+D_l}{2}\right)} \quad \text{……………………………..…………………..……….} \quad (5.2)$$

where, α_{3D} is a 3D density term, Γ( ) is a gamma function, D_{c3D} and D_{l3D} are 3D fractal dimensions of fracture center and length distribution, respectively. These parameters are equal to D_c+1 and D_l+1.

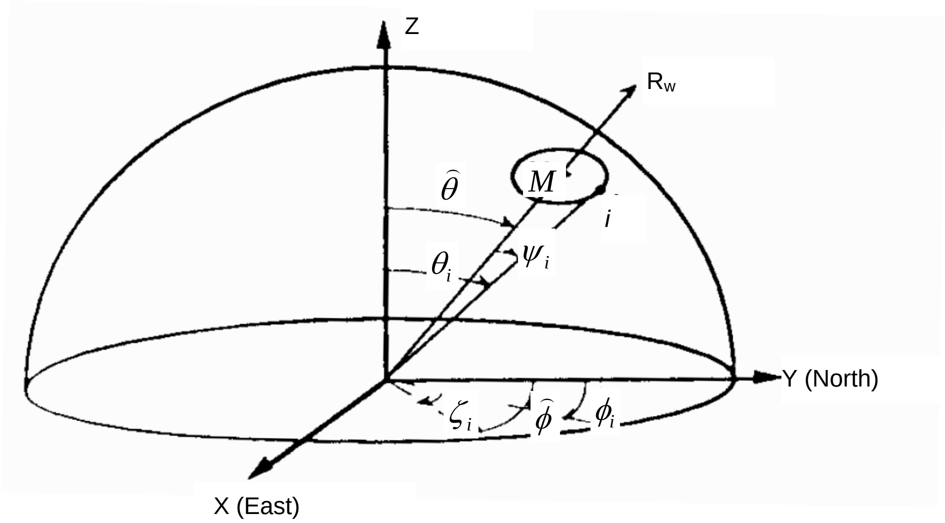


Fig. 5.1-3D expression of *i*th fracture orientation (From Villaescusa[32]).

5.1.2 Fracture Orientation in 3D Space

Compared to the orientation in 2D FDFN expressed by strike, i^{th} fracture orientation in 3D is expressed by dip, θ_i, and dip direction, ϕ_i, which are used to calculate normal vector components of fractures **(Fig. 5.1)**. In Fig. 5.1, *M* is a mean unit vector for which dip and dip direction are $\widehat{\theta}$ and $\widehat{\phi}$, respectively. ψ_i and ζ_i are angular and azimuthal distances between the mean and i^{th} fracture[32]. In this research, 3D fracture orientations are generated by the method suggested by Villaescusa[32]. Dip and dip direction of i^{th} fracture are generated along the mean unit vector, *M* within the boundaries of angular distance and azimuthal distance which are calculated by Eq. 5.3 and 5.4.

$$\zeta_i = 2\pi N(0,1) \quad \text{(5.3)}$$

$$\cos\psi_i = 1 + \frac{1}{K}\ln\{1 - N(0,1)\} \quad \text{(5.4)}$$

Fracture orientation generation consists of two coordinate rotations. The first step is calculating normal vector components of i^{th} fracture in a local coordinate system where Z-direction is parallel to the mean orientation **(Fig. 5.2)**. Normal vector components (*u*, *v*, *w*) of i^{th} fracture are calculated by Eq. 5.5.

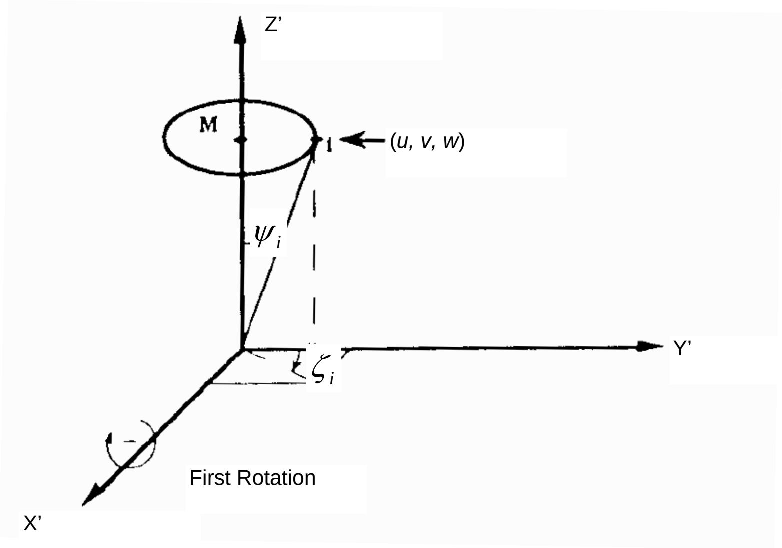


Fig. 5.2-Normal vector components of i^{th} fracture in a local coordinate system (From Villaescusa[32]).

$$\begin{aligned} u &= \sin\psi_i \sin\zeta_i \\ v &= \sin\psi_i \cos\zeta_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5.5) \\ w &= \cos\psi_i \end{aligned}$$

Then, the coordinate system in Fig. 5.2 is rotated along the X'-axis by the mean dip, $\widehat{\theta}$. After the first rotation, the Z-axis is the same as the original Z-axis in Fig. 5.1. The new coordinates of normal vector components (*u*, *v'*, *w'*) are calculated by Eq. 5.6 and the new system is shown in **Fig. 5.3**.

$$\begin{aligned} u &= \sin\psi_i \sin\zeta_i \\ v' &= v\cos\widehat{\theta} + w\sin\widehat{\theta} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5.6) \\ w' &= -v\sin\widehat{\theta} + w\cos\widehat{\theta} \end{aligned}$$

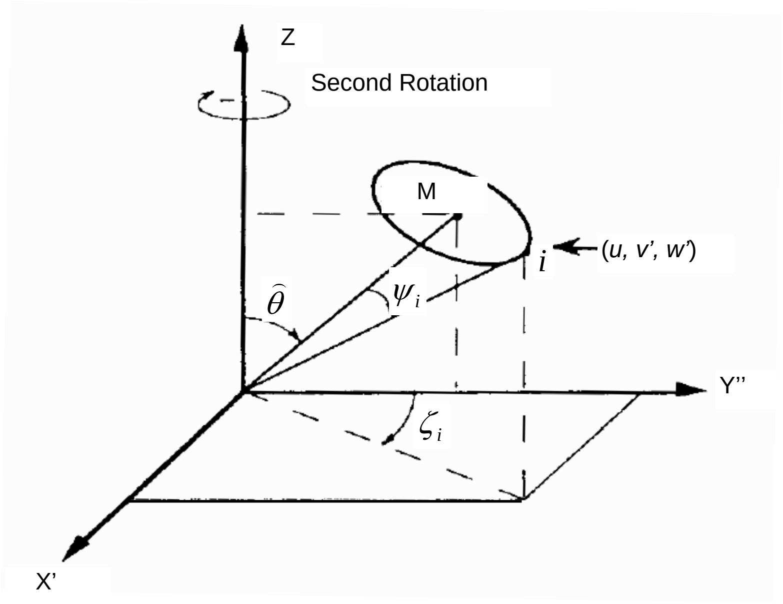


Fig. 5.3-New normal vector coordinates after the first rotation (From Villaescusa[32]).

The second rotation is done along the Z-axis by the mean dip direction, $\widehat{\phi}$. After the second rotation, all the axes are matched with the original axes. The final coordinate of normal vector components (u', v'', z') of i^{th} fracture are calculated by Eq. 5.7. The final geometry is shown in **Fig. 5.4**.

$$\begin{aligned} u' &= u\cos\widehat{\phi} + v'\sin\widehat{\phi} \\ v'' &= -u\sin\widehat{\phi} + v'\cos\widehat{\phi} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (5.7) \\ w' &= -v\sin\widehat{\theta} + w\cos\widehat{\theta} \end{aligned}$$

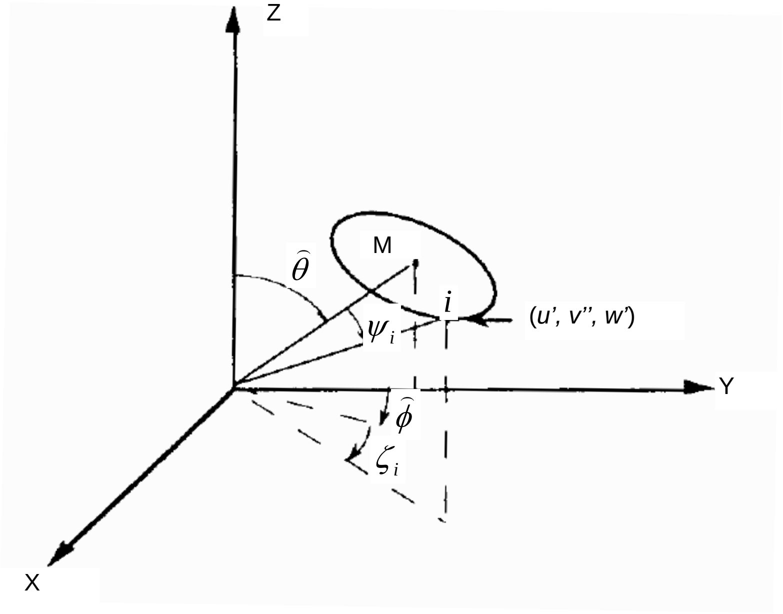


Fig. 5.4-Final geometry of the normal vector of *i*th fracture in the global coordinate system (From Villaescusa[32]).

5.1.3 Fracture Center and Aperture Generation in 3D Space

Fig. 5.5 and 5.6 show the sample figures of fracture center distributions and FDFN in the 3D space using a multiplicative cascade process. Fracture shape in 3D space is uncertain, so some assume disc shape[28, 31, 33], and others assume polygons. In this research, fracture shape in 3D is assumed a disc because disc shape is easy to generate and efficient in computing fracture geometry. Black lines in Fig. 5.6 represent wells. While 1D profile types of fracture aperture (see Fig. 4.4) are combined with 2D FDFN, 2D aperture fields are combined with 3D FDFN. **Fig. 5.7** is a contour map of aperture distribution generated by a corrected SRA, and this type of aperture distributions are to be assigned to each fracture. Aperture size increases by the order of blue, yellow, and red.

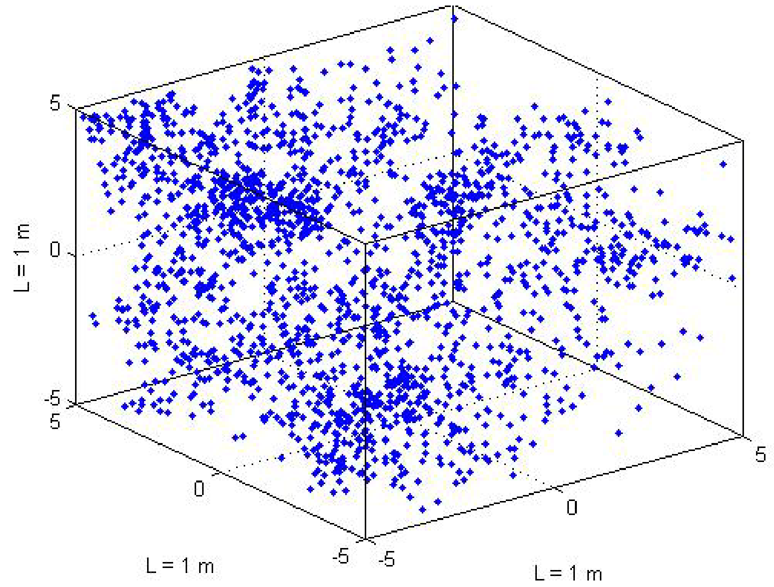


Fig. 5.5-Fracture center distribution in 3D. Clustering is clearly shown.

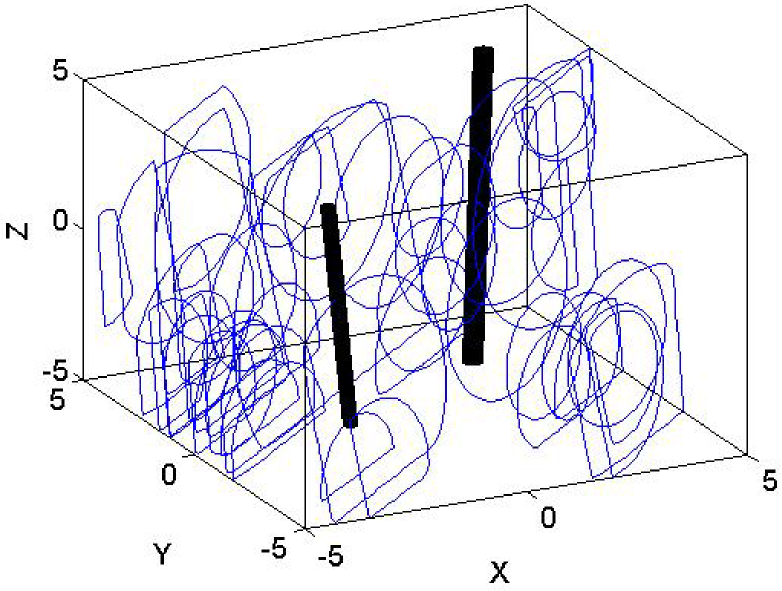
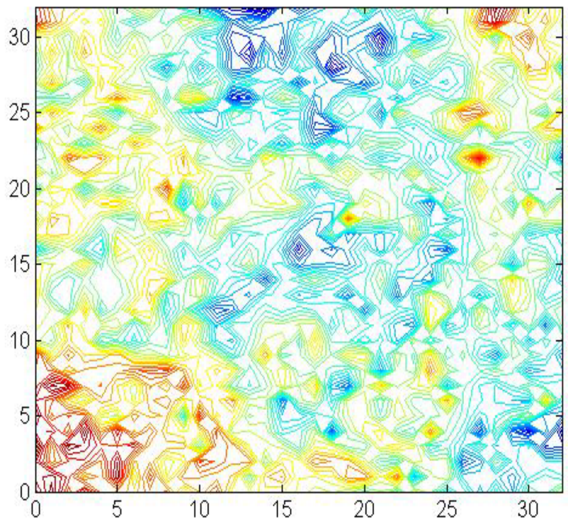
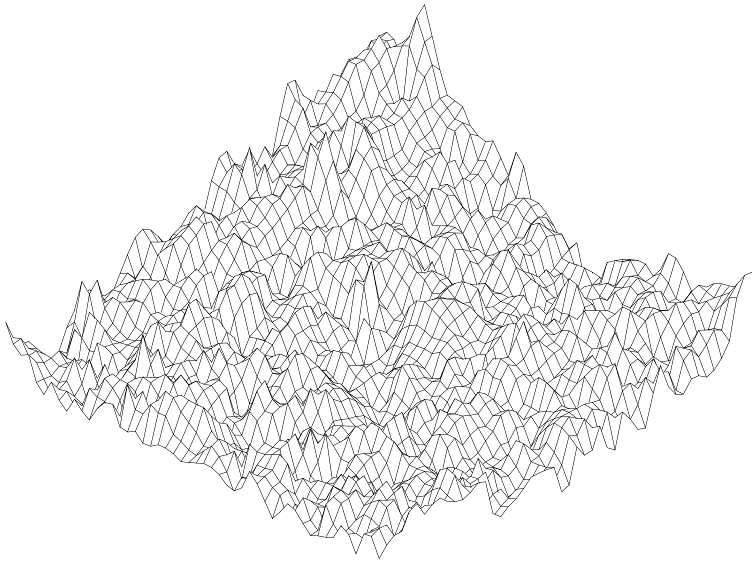


Fig. 5.6-Sample 3D FDFN with two wells (dimensions are in meters). Fracture shape is assumed as a disc, and black lines represent wells.



a. Contour map

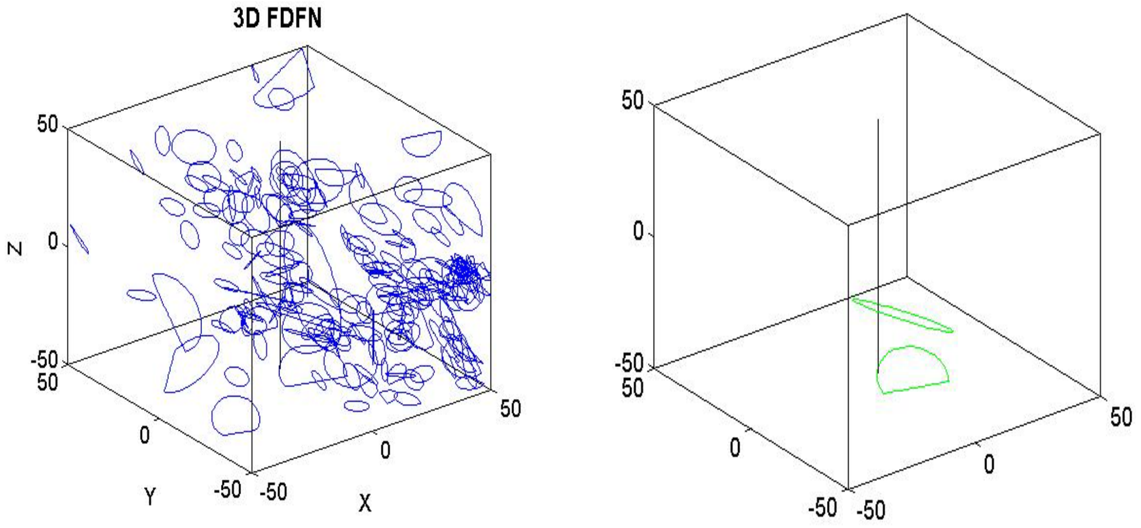


b. Surface map

Fig. 5.7-2D fractal fracture aperture. Aperture increases by the order of blue, yellow and red. Dimensions of X and Y axes are dimensionless.

5.1.4 General Features of the 3D FDFN Generation Codes

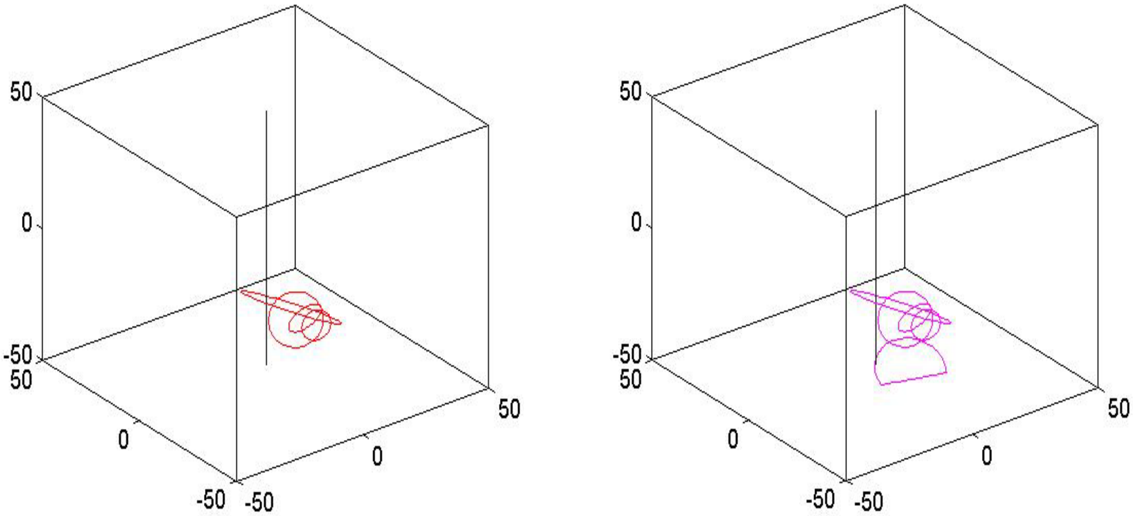
Basically 2D and 3D FDFN generation codes are similar. Increasing Euclidean dimension allows more options such as well inserting (Fig. 5.6), and well communicated fracture detection. **Fig. 5.8** shows the well communicated fracture detecting procedures. First, one detects fractures directly connected to a well (Fig. 5.84b). Then, one searches for other fractures connected to well-connected fractures (Fig. 5.8c). Finally, all fractures communicating to well are detected and the volume of those fractures is calculated (Fig. 5.8d). Another distinctive option of the 3D FDFN generation codes is a section slicing option, which allows for a cutting view of 3D FDFN. This helps to analyze generated 3D FDFN using 2D theories. In addition, 3D FDFN generation codes have Monte Carlo simulation and fractal fracture porosity estimation options as well.



a. 3D FDFN

b. Detecting fractures directly connected to a well

Fig. 5.8-Well communicating fracture detection procedures.



c. Detecting fractures connected to well-connected fractures   d. All fractures communicating to a well

Fig. 5.8-Continued.

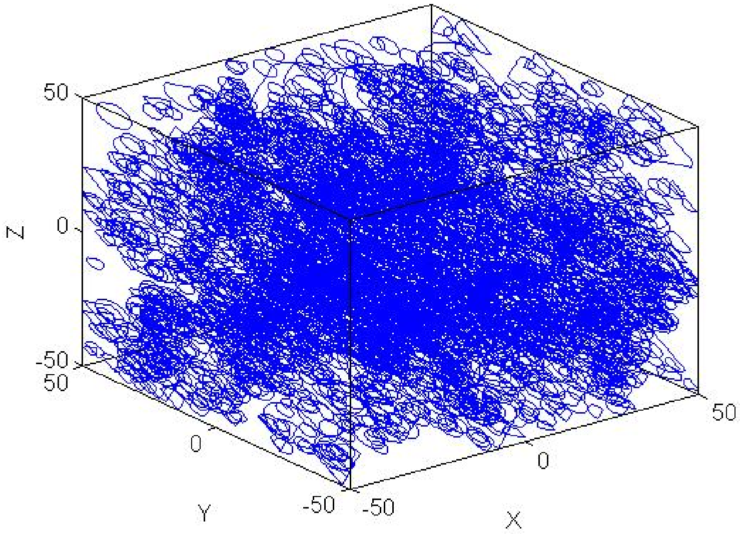
5.2 Validation of 3D FDFN Generation Codes

5.2.1 Self vs. Self Validation

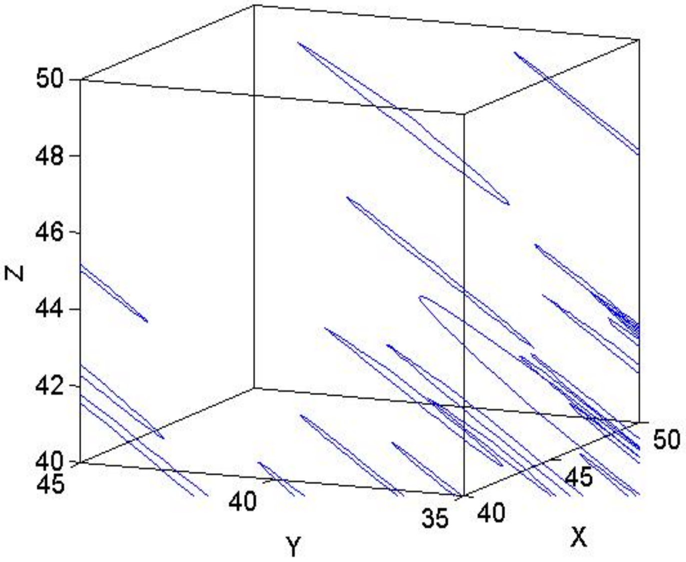
Validation of 3D FDFN generation codes is performed in two folds as well. The first one is self versus self validation by comparing input with the fractal analysis results of generated 3D FDFN. The next validation is done by comparing the fractal characteristics of generated 3D FDFN with theoretical models. **Fig. 5.9 and 5.10** show generated 3D FDFN and the results of fractal analysis. Fig. 5.9b shows part of the generation domain to illustrate fractures in detail. Input fractal dimensions of center and length distributions are 1.6 and 1.8, respectively. The value of a density term is 2.5 which is equivalent to 5.723 in 3D density term, α_{3D}. The total number of fracture generated is 3576. By using Eq. 4.1 and 4.2, fractal dimensions of center and length distributions and 3D density term are calculated (see Fig. 5.10). Fractal dimension of center, D_{c3D} is 2.577, fractal dimension of length, D_{l3D} is 2.803, and 3D density term, α_{3D} is 6.415. These parameter values are very close to input and differences are acceptable in FDFN realizations.

Another validation was performed by analyzing 2D sections of 3D FDFN. Nine sections were obtained, each three sections is perpendicular to x, y, and z axes, respectively. **Fig.**

5.11 shows one of nine sections and **Table 5.1** lists the results of analysis. Input values of D_{c3D} and D_{l3D} are 2.6 and 2.8. Fractal dimensions of center tend to higher than the input about 15%. Fractal dimensions of length are much similar to the input. However, both parameters of nine sections are close enough to the input.

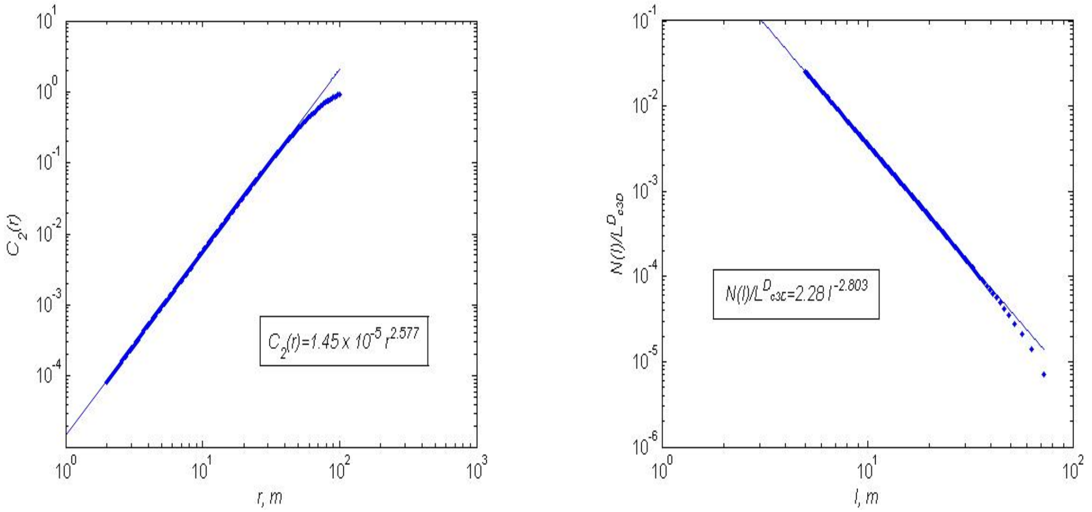


a. 3D FDFN



b. Zoomed view of 3D FDFN

Fig. 5.9-3D FDFN with 3576 fractures (dimensions are in meters).



a. Fracture center distribution b. Fracture diameter distribution

Fig. 5.10-Fractal analysis of 3D FDFN shown in Fig. 5.9.

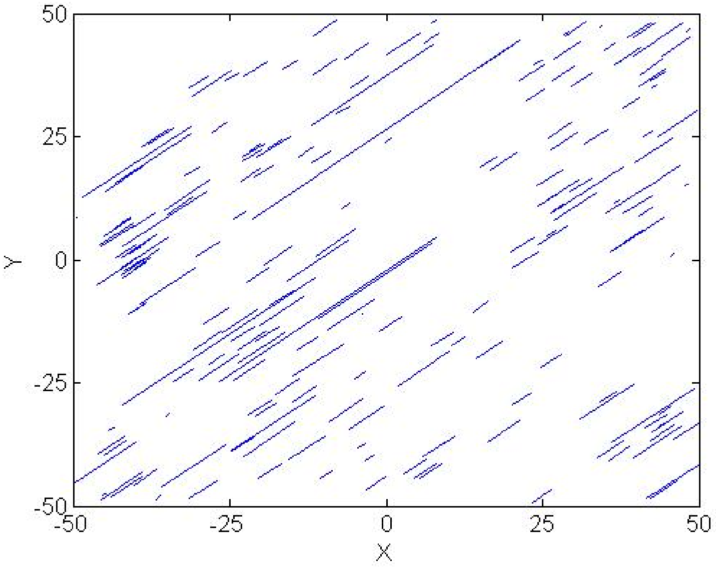


Fig. 5.11-One of nine 2D trace maps of 3D FDFN (dimensions are in meters).

Table 5.1 Results of Fractal Analysis of Nine Sections

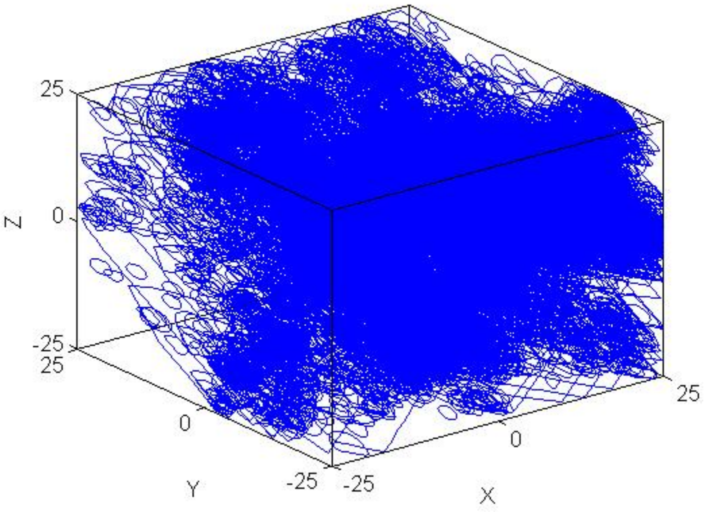
Plane	XY			YZ			ZX			Avg.
D_c	1.865	1.845	1.768	1.884	1.865	1.876	1.777	1.829	1.849	1.840
D_l	1.872	1.790	1.810	2.173	1.904	1.785	1.768	1.907	1.969	1.886

5.2.2 Validation Using Theoretical Models

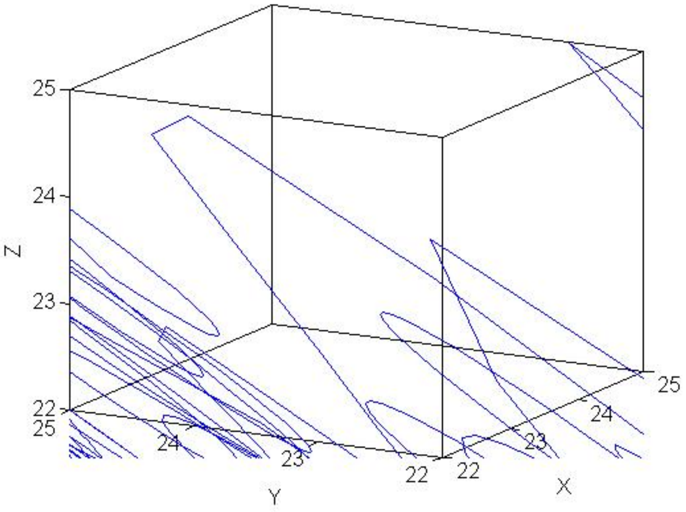
Using Eq. 4.10, geometry of 3D FDFN is validated by analyzing 24 different cases. Eq. 4.10 deals with fracture length rather than diameter or other 3D size parameters, so that 2D sections of 3D FDFN were used in analysis. **Fig. 5.12** shows one case of 3D FDFN and **Fig. 5.13** shows its fracture trace map and fracture distance analysis. To obtain enough number of fracture traces, 2D sections were perpendicular to z-axis and selected at the center of the 3D domain. The theoretical value of the case shown in Fig 5.12 and 5.13 is 0.7368 and the difference is just 3.13%. The analysis results are shown in **Fig. 5.14**. The bold lines represent the theoretical values. Analysis well shows that generation results are matched with the theoretical values, regardless of input. The maximum difference is only 13.6% and such error is small enough to be accepted in synthetic fracture network generation. Therefore, 3D FDFN generation codes are validated and can be said to produce fracture networks which well reflect characteristics of real fracture networks.

Fracture porosity estimated by 3D FDFN generation codes was validated by using Eq. 4.11. Six different cases were tested. Size of generation domain started from 10 m by 10 m and increase 10% of its side length as well. At each domain size, 5 different generations were done and the average fracture porosity was calculated. In 3D validation of fracture porosity, 40 different domain sizes were tested while 50 different domain sizes were tested in 2D due to the limitation of memory and long generation time of 3D FDFN. **Table 5.2** lists input data of fractal dimensions of center and length distributions, and fractal dimension of fracture porosity obtained by regression and R^2 values. R^2 values show well that fractal dimensions of fracture porosity estimated from 3D FDFN are well matched with the mathematical model (Eq. 4.11). **Fig. 5.15** shows the interesting trends as Fig 4.16

shows. With increasing fractal dimension of fracture center distribution, D_{c3D}, fractal dimension of fracture porosity, D_{fp3D} tends to decrease. On the other hand, D_{fp3D} decreases as D_{l3D} increases. Increasing D_{c3D} causes the increase of the number of fractures. It means that fracture porosity with high D_{c3D} at a larger domain is larger than the value with low D_{c3D}. However, high D_{l3D} causes the generation of short fractures so that fracture porosity decreases.

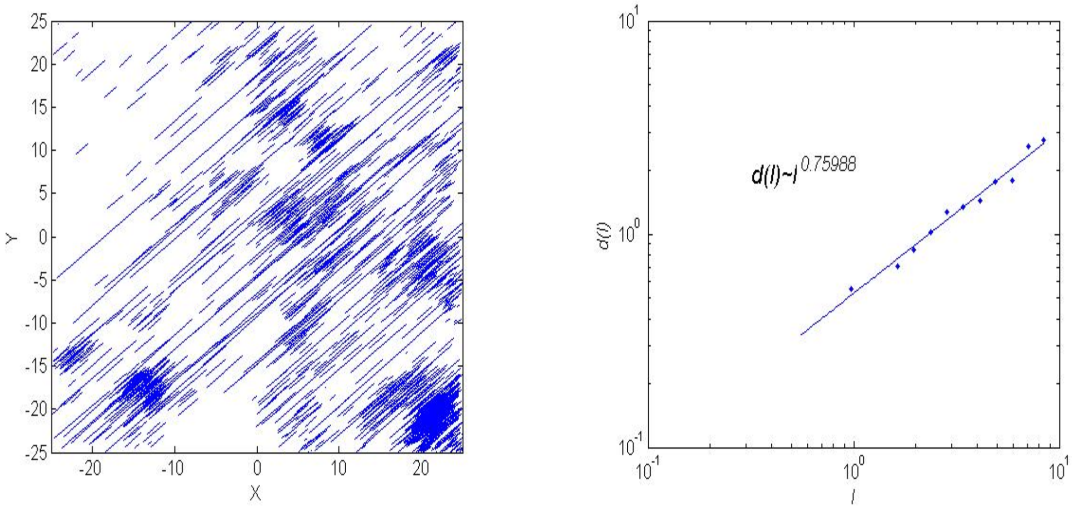


a. 3D FDFN



b. Zoomed view of 3D FDFN

Fig. 5.12-Sample generation of 3D FDFN (D_c=1.9 and D_l=1.4). All dimensions are in meters.



a. Fracture trace map b. Fracture distance

Fig. 5.13-2D section and its fractal analysis result of fracture distance (dimensions are in meters).

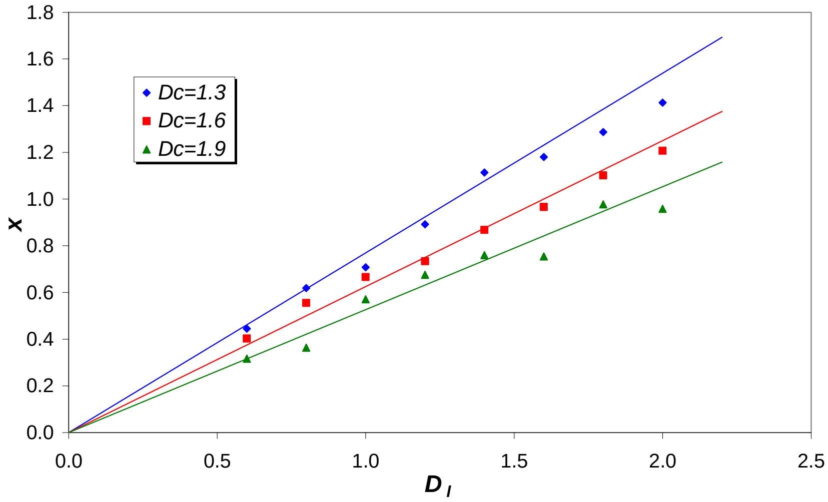


Fig. 5.14-Comparison between theoretical values and the fractal analysis results of 3D FDFN.

Table 5.2 Input Data and R^2 Values of the 3D Cases

	D_{c3D}	D_{l3D}	D_{fp3D}	R^2
Case 1	2.3	2.4	1.8884	0.9995
Case 2	2.3	2.8	1.4434	0.9995
Case 3	2.6	2.4	2.2322	0.9986
Case 4	2.6	2.8	1.7883	0.9997
Case 5	2.9	2.4	2.5497	0.9943
Case 6	2.9	2.8	2.1036	0.9994

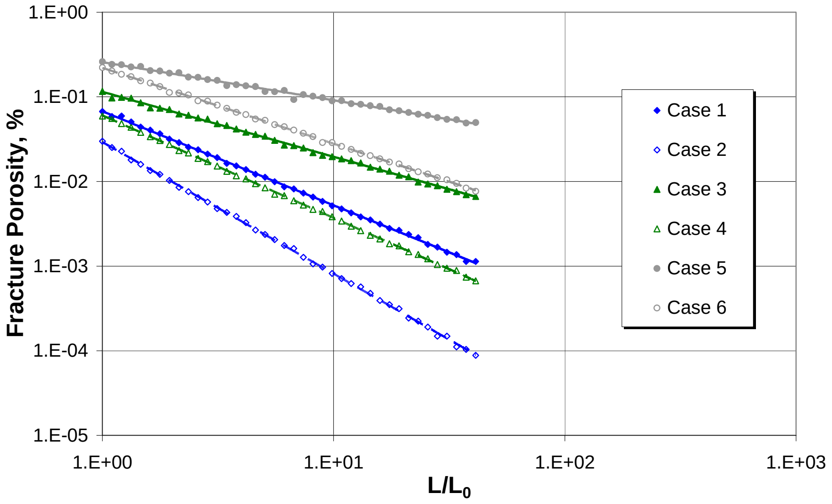


Fig. 5.15-Fractal behavior of fracture porosity. Bold lines denote the trend lines of cases 1, 3, 5, and dashed lines denote the trend lines of cases 2, 4, 6.

CHAPTER VI

FRACTURE POROSITY ESTIMATION USING 2D AND 3D FDFN GENERATION CODES

6.1 Efficient Means of Fracture Porosity Estimation

6.1.1 Comparison of 2D and 3D Fractal Dimensions of Fracture Porosity Estimation

Fractal dimension of fracture porosity, D_{fp} values obtained from 2D and 3D realizations were compared. This comparison used twelve different cases, six from 2D and six from 3D realizations (see Table 4.3 and 5.2). **Fig. 6.1** shows the result of the comparison. For the characteristics of fractal dimensions, 2D fractal dimension values added by one are equivalent to 3D fractal dimensions. As shown in the figure, 2D and 3D fractal dimension values are almost identical. These values distribute around the 45° line which represents the perfect match. Thus, it can be said that 2D and 3D realizations produced the same results in estimating fractal dimension of fracture porosity. This result shows that 2D realizations are superior to 3D realizations in that 2D realizations are much faster than 3D. It can be concluded that 2D realizations are more efficient than 3D realizations when estimating fractal dimension of fracture porosity.

6.1.2 Comparison of 2D and 3D Fracture Porosity Estimation

Fracture porosity estimated by 2D and 3D realizations was compared to find more efficient ways of fracture porosity estimation. Before determining the most efficient means of fracture porosity estimation, characteristics of fracture porosities estimated by 2D and 3D realizations were compared. **Fig 6.2** compares the six different cases (refer Table 4.3 and 5.2 for detailed input values). Squares and dots represent 2D and 3D cases, respectively. Case 1 and 2, 3 and 4, and 5 and 6 share the same values of fractal dimension of fracture center distributions. Fractal dimensions of fracture length distribution for case 1 and 2 are 2.4 and 2.8, respectively. Fracture porosity estimated by 2D realization is a kind of average porosity per unit thickness. Thus, 2D realization assumes that generated fractures are prolonged in the thickness direction. It means that fracture porosity estimated by 2D realizations tends to be higher than the values estimated by 3D realizations, if enough numbers of fractures are generated (Fig. 6.2a-d). Fracture porosity differences between 2D and 3D realizations are narrowing and 3D fracture porosity estimation is higher than 2D estimation with increasing fractal dimensions of fracture center distribution. Such 3D fracture porosity reversion is caused by an increase in the number of fractures in 3D realizations compared to the fracture number increase in 2D cases **(Fig. 6.3)**. Thus, the fractal porosity values estimated by 3D realizations are higher than those estimated by 2D realizations (Fig. 6.2e-f). The differences of fracture porosity of odd number cases are less than those of even number cases, because increasing fractal dimensions of fracture length distribution reduce fracture number differences between 2D and 3D realizations **(Fig. 6.4)**.

6.1.3 Two Step Procedure of Fracture Porosity Estimation

Fractal dimension of fracture porosity is not affected by the Euclidean dimension of FDFN realizations. Both 2D and 3D realizations are good at the estimation of fractal dimension of fracture porosity, but 3D realizations take much longer time than 2D realizations. Thus, 2D realizations are the appropriate method in the estimation of fractal dimension of fracture porosity. On the other hands, estimated fracture porosity values are affected by the Euclidean dimension. The assumption of 2D realizations, in which fractures are prolonged in the thickness direction, is far from the real nature. Thus, 3D realizations are superior to 2D realizations in the estimation of fracture porosity values. Therefore, the effective way of fracture porosity estimation consists of two steps. The first step is to estimate the fractal dimension of fracture porosity using 2D FDFN generation codes. The next step is to estimate ϕ_{f0} (initial fracture porosity with domain size L_0) of Eq. 4.11 using 3D FDFN generation codes. Through two steps, two important parameter values of Eq. 4.11 are estimated. Thus, fracture porosity of any domain size, L can be estimated.

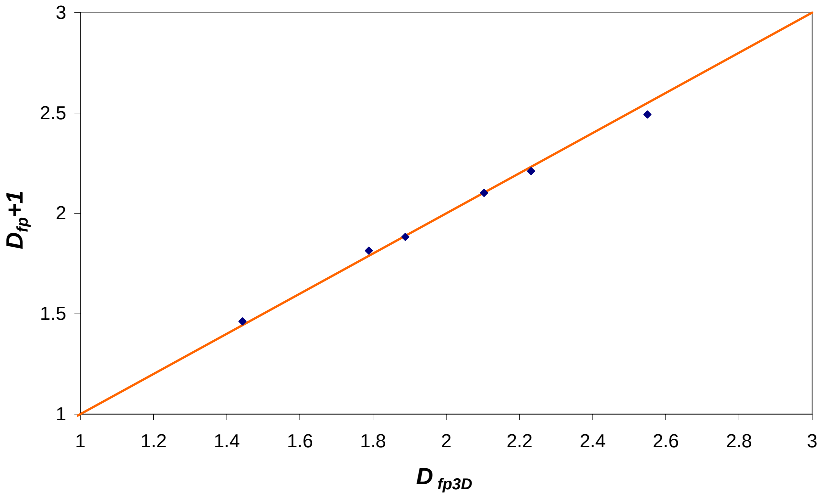
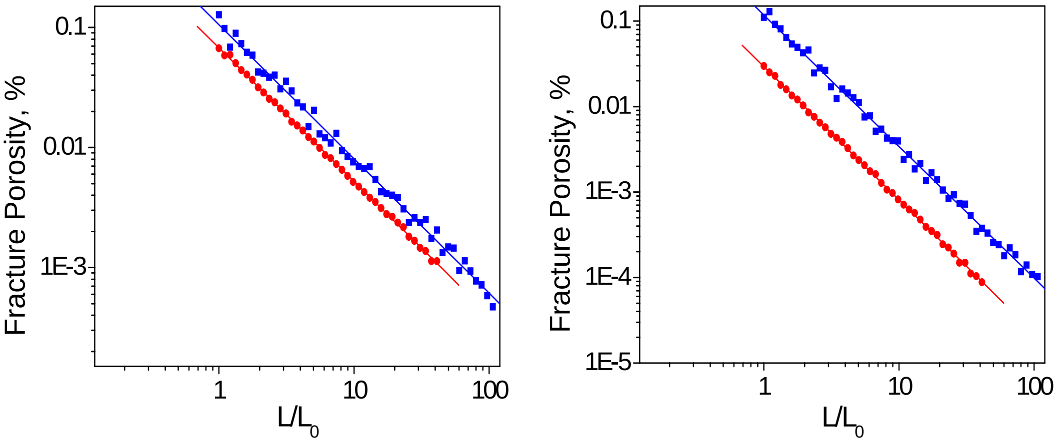


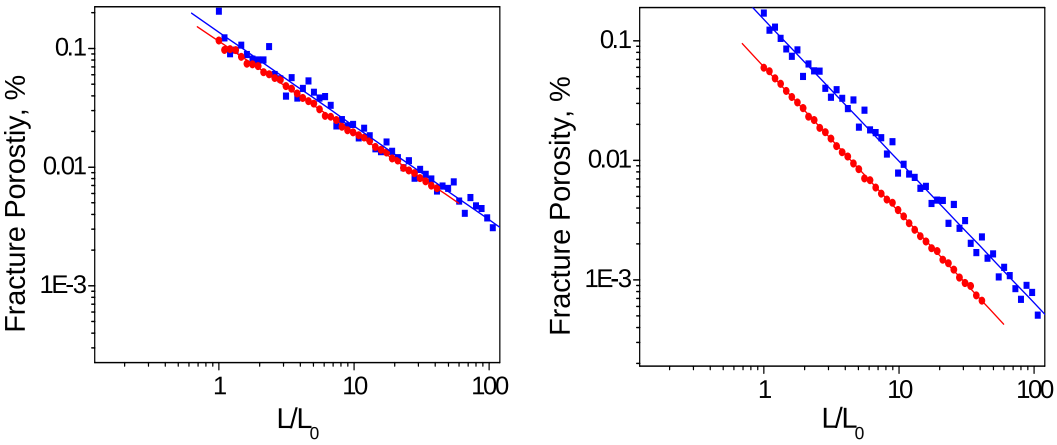
Fig. 6.1-Comparison of 2D and 3D fractal dimensions of fracture porosity.



a. Case 1

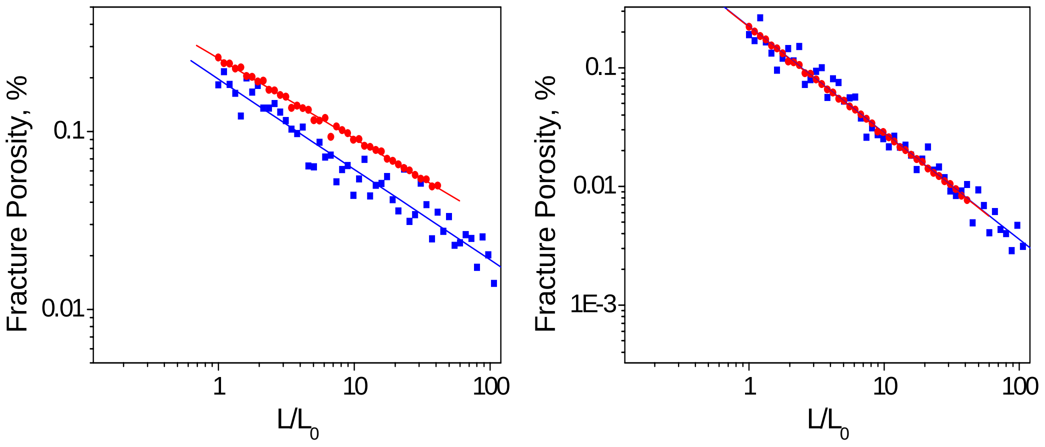
b. Case 2

Fig. 6.2-Comparison of estimated fractal porosity of various cases. Squares and dots represent 2D and 3D, respectively.



c. Case 3

d. Case 4



e. Case 5

f. Case 6

Fig. 6.2-Continued.

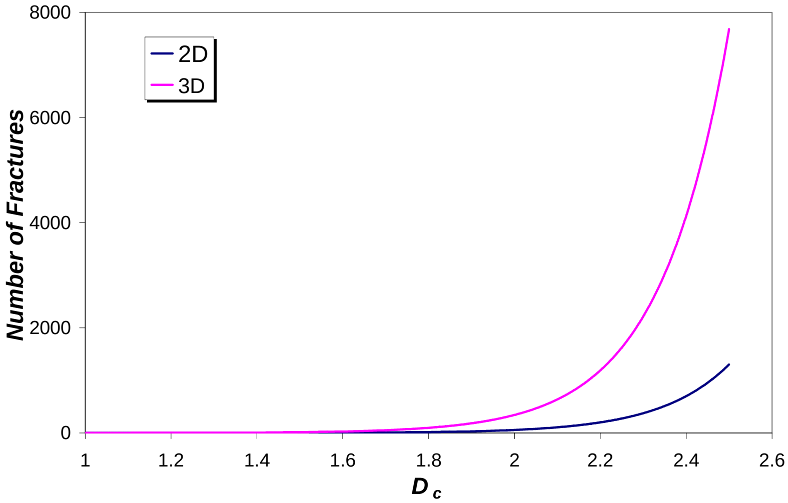


Fig. 6.3-Number of fractures as a function of fractal dimension of fracture center distribution.

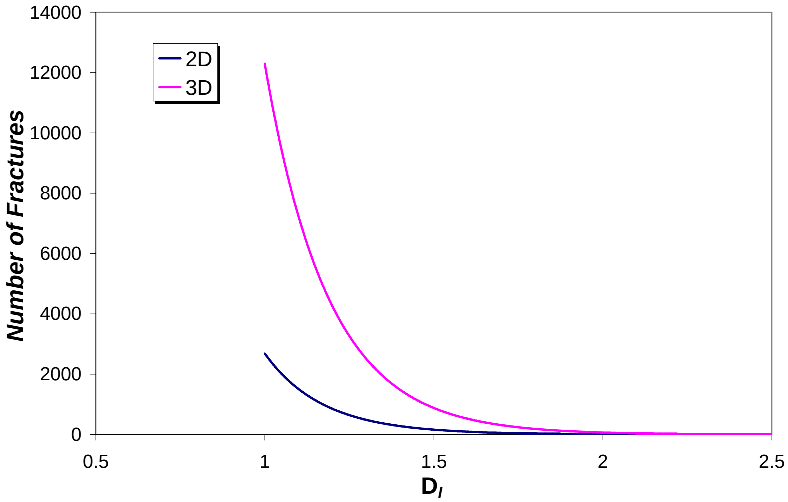


Fig. 6.4-Number of fractures as a function of fractal dimension of fracture length distribution.

CHAPTER VII

CONCLUSIONS

Experimental study of artificial fracture using X-ray CT scanner gave the following results:

1. Hydrological and mechanical properties of fracture depend on aperture rather than the roughness of one surface.
2. Properties of the aperture are dependent on scale and confining pressure. It is shown that a representative size of relatively smooth planar fracture depends not only on scale but also on confining pressure. Although fractal dimension and amplitude parameter do not reach the stationary threshold due to the small size of the specimen, both parameters fluctuated within a band, and within a certain size of sampling area (between 1500 and 2400 mm^2). Confining pressure affects both parameters, so each parameter differs under different confining pressures for a reference sampling size. Therefore, in order to characterize fracture aperture distribution, a sufficient specimen size should be obtained. Also, in situ confining pressure should be taken into consideration.
3. Utilizing the X-Ray CT Scanner enables researchers to overcome the limitation of examining one surface of a rock fracture. While a 3-D laser scanner investigates the roughness of only one surface, the calibration technique of CT scans measures aperture distribution, which is the result of interactions between two rock surfaces.
4. Fracture deformation behavior and normal fracture stiffness were also calculated by the novel method. Conventional fracture property experiment methods cannot obtain mechanical aperture distribution, fracture stiffness, and effluent data at the same time. Alternatively, the X-ray CT scanner makes it possible to amass these data simultaneously.

The results of the experimental study of real fractures using X-ray CT scanner may be summarized as follows:

1. Fractal dimension and amplitude of the natural fracture also change with sample size.
2. Confining pressure was not applied, but natural fracture may show the same tendency with different confining pressure by analogy.

The main conclusions of FDFN generation codes development and numerical study follow:

1. Using fractal theories, FDFN generation codes have been developed to estimate fracture porosity of Type 1 NFR.
2. Developed FDFN generation codes are compared to the theoretical models which are not applied in algorithms for the verification. Generated FDFN are well matched with the theoretical model.
3. Fractal dimensions of fracture porosity estimated using 2D and 3D FDFN generation codes are almost identical. Due to rapid running time, 2D FDFN generation codes are superior to 3D codes in estimating fractal dimension of fracture porosity.
4. 3D FDFN generation codes are good at estimating fracture porosity with any domain size.
5. The most efficient way of estimating fracture porosity is combining 2D and 3D FDFN generation codes. 2D codes are used for estimating fractal dimension of fracture porosity, and 3D codes are used for estimating the initial fracture porosity. This two step procedure can estimate fracture porosity regardless of domain size.

6. The developed FDFN generation codes enable researchers to utilize any available data such as outcrop maps, FMI logging, CT images and/or core fractures to estimate fracture porosity.

NOMENCLATURE

A = amplitude, mm

$C_2(\ )$ = correlation function used to calculate a fractal dimension of fracture center distribution

D = fractal dimension in general

D_c = fractal dimension of fracture center distribution in 2D

D_{c3D} = fractal dimension of fracture center distribution in 3D ($D_{c3D}= D_c +1$)

D_{fp} = fractal dimension of fracture porosity in 2D

D_{fp3D} = fractal dimension of fracture porosity in 3D ($D_{fp3D}= D_{fp} +1$)

D_l = fractal dimension of fracture length distribution in 2D

D_{l3D} = fractal dimension of fracture length distribution in 3D ($D_{l3D}= D_l +1$)

E = embedded Euclidean dimension

H = Hurst exponent

$IntCT$ = integrated CT signal

K = Fisher's constant

K_n = normal fracture stiffness, MPa/μm

L = side length of a generation domain, m

$N(L)$ = number of fractures whose length is longer than l_{min}

$N_p(r)$ = number of points whose distance is less than r

P_i = i^{th} probability

$R^i_{F,K}$ = fracture angular deviation from the mean, degree

$R^i_{G,1}$ = Gaussian random number between 0 and 1

$S(w)$ = standard deviation of asperity height when sampling widow length is w

c_f = fracture compressibility, MPa^{-1}

$d(l)$ = distance between a fracture and its nearest neighbor having a length larger than l, m

h = lag or height of aperture, μm

l = fracture length, m

l_{min} = minimum fracture length, m

q = degree of multifractal

sr = scale ratio

w = sampling window length, mm

x = exponent, D_l/D_c

$\Gamma(\,)$ = gamma function

α = fracture density in 2D

α_{3D} = fracture density in 3D

δ = displacement, μm

σ = normal stress, MPa

ϕ_f = fracture porosity, fraction

ϕ_{f0} = initial fracture porosity with domain size L_0, fraction

REFERENCES

1. Nelson, A.R.: *Geologic Analysis of Naturally Fractured Reservoirs*, Gulf Professional Publishing, Boston (2001).
2. Pyrak-Nolte, L. *et al*.: “Hydraulic and Mechanical Properties of Natural Fractures in Low Permeability Rock,” *Proc.*, 6th Congress of the ISRM, Montreal, Canada (1987) 225.
3. Myer, L.: “Hydromechanical and Seismic Properties of Fractures,” *Proc.*, 7th Congress of the ISRM, Aachen, Germany (1991) 397.
4. Keller, A.: “Single and Multiphase Flow and Transport in Fractured Porous Media,” Ph. D. dissertation, Stanford University, Stanford, California (1996).
5. Hakami, E. and Larsson, E.: “Aperture Measurements and Flow Experiments on a Single Natural Fracture,” *Int. J. Rock Mech. & Min. Sci.* (1996) **33,** No. 4, 395.
6. Lanardo, F., Jing, L., and Stephansson, O.: “Scale Dependency of Roughness and Stationarity of Rock Joints,” *Proc.*, 9th Cong. of the ISRM, Paris, France (1999) 1391.
7. Fardin, N, Stephansson, O., and Jing, L.: “The Scale Dependence of Rock Joint Surface Roughness,” *Int. J. Rock Mech. & Min. Sci.* (2001) **38,** No. 1, 659.
8. Bertel, S. and DiCarlo, D.: “Measurement of Aperture Distribution, Capillary Pressure, Relative Permeability, and In Situ Saturation in a Rock Fracture Using Computed Tomography Scanning,” *Water Resour. Res.* (2001) **37,** No. 3, 649.

9. Muralidharan, V. *et al*.: “Investigating Fracture Aperture Distribution Under Various Stress Conditions Using X-Ray CT Scanner,” paper CIPC 2004-230 presented at the 2004 Annual Technical Meeting of the Petroleum Society, Calgary, Canada, 8-10 June.
10. Konzuk, J.S. and Kueper, B.H.: “Evaluation of Cubic Law Based Models Describing Single-Phase Flow Through a Rough-Walled Fracture,” *Water Resour. Res.* (2004) **40**, W02402.
11. Barton, C.C. and Larsen, E.: “Fractal Geometry of Two-Dimensional Fracture Networks at Yucca Mountain, South-Western Nevada,” *Proc.*, Intl. Symposium on Fundamentals of Rock Joints, Bjorkliden, Sweden (1985) 77.
12. Okubo, P.G. and Aki, K.: “Fractal Geometry in the San Andreas Fault System,” *J. Geophys. Res.* (1987) **92,** 345.
13. Hirata, T.: “Fractal Dimension of Fault Systems in Japan: Fractal Structure in Rock Fracture Geometry at Various Scales,” *Pure & Applied Geophysics* (1989) **131**, 157.
14. Sakellariou, M., Nakos, B., and Mitsakaki, C.: “On the Fractal Character of Rock Surfaces,” *Int. J. Rock Mech. & Min. Sci.* (1991) **28,** No. 6, 527.
15. Odling, N.E.: “Network Properties of a Two-Dimensional Natural Fracture Pattern,” *Pure & Applied Geophysics* (1992) **138**, No. 1, 95.
16. Gillespie, P.A. *et al.*: “Measurement and Characterisation of Spatial Distributions of Fractures,” *Tectonophysics* (1993) **226,** 113.
17. Ghosh, A. and Daemen, J.J.K.: “Fractal Characteristics of Rock Discontinuities,” *Engineering Geology* (1993) **34,** 1.

18. Ledesert, B. *et al.*: "Fractal Analysis of Fractures Applied to Soultz-Sous-Forets Hot Dry Rock Geothermal Program," *J. Volcanology & Geothermal Res.* (1993) **57,** 1.
19. Kulatilake, P.H.S.W., Wathugala, D.N., and Stephansson, O.: "Joint Network Modelling with a Validation Exercise in Stripa Mine, Sweden," *Int. J. Rock Mech. Min. Sci.& Geomech. Abstr.* (1993) **30,** No. 5, 503.
20. Gillespie, P.A. *et al.*: "Measurement and Characterisation of Spatial Distributions of Fractures," *Tectonophysics* (1993) **226**, 113.
21. Watanabe, K. and Takahashi, H.: "Fractal Geometry Characterization of Geothermal Reservoir Fracture Networks," *J. Geophys. Res.* (1995) **100,** No. B1, 521.
22. Bour, O. and Davy, P.: "Connectivity of Random Fault Networks Following a Power Law Fault Length Distribution," *Water Resour. Res.* (1997) **33**, No. 7, 1567.
23. Ehlen, J.: "Fractal Analysis of Joint Patterns in Grinite," *Int. J. Rock Mech. & Min. Sci.* (2000) **37,** 909.
24. Babadagli, T.: "Fractal Analysis of 2-D Fracture Networks of Geothermal Reservoirs in South-Western Trukey," *J. Volcanology & Geothermal Res.* (2001) **112,** 83.
25. Rawnsley, K. and Wei, L.: "Evaluation of a New Method to Build Geological Models of Fractured Reservoirs Calibrated to Production Data," *Petroleum Geoscience* (2001) **7**, 23.
26. Bour, O., Davy, P., and Darcel, C.: "A Statistical Scaling Model for Fracture Network Geometry, with Validation on a Multiscale Mapping of a Joint Network (Hornelen Basin, Norway)," *J. Geophys. Res.* (2002) **107,** No. B6, 2113, 10.1029/2001JB000176.

27. Darcel, C. *et al.*: “Connectivity Properties of Two-Dimensional Fracture Networks with Stochastic Fractal Correlation,” *Water Resour. Res.* (2003) **39**, No. 10, 1272.
28. Piggott, A.R.: “Fractal Relations for the Diameter and Trace Length of Disc-Shaped Fractures,” *J. Geophys. Res.* (1997) **102,** No. B8, 18121.
29. Luthi, S.M. and Souhaite, P.: “Fracture Apertures from Electrical Borehole Scans,” *Geophysics* (July 1990) **55,** No. 7, 821.
30. Gale, J.E.: “Comparison of Coupled Fracture Deformation and Fluid Flow Models With Direct Measurements of Fracture Pore Structure and Stress-Flow Properties,” *Proc.,* the 28th U.S. Rock Mechanics Symposium, Tucson, Arizona (1987) 1213.
31. Priest, S.D.: *Discontinuity Analysis for Rock Engineering,* Chapman & Hall, London (1992).
32. Villaescusa, E.: “Statistical Modelling of Rock Jointing,” *Proc.,* the Conference on Probabilistic Methods in Geotechnical Engineering, Rotterdam, Netherlands (1993) 221.
33. Tamagawa. T. *et al.*: “Construction of Fracture Network Model Using Static and Dynamic Data,” paper SPE 77741 presented at the 2002 SPE Annual Technical Conference and Exhibition, San Antonio, Texas, 29 September-2 October.
34. Wellington, S.L. and Vinegar, H.J.: “CT Studies of Surfactant-Induced CO_2 Mobility Control,” paper SPE 14393 presented at the 1985 SPE Annual Technical Conference and Exhibition, U.S.A., 22-25 September.
35. Feder, J.: *Fractals,* Plenum, New York, (1988).

36. Kulatilake, P.H.S.W. and Um, J.: “Requirements for Accurate Quantification of Self-Affine Roughness Using the Roughness-Length Method,” *Int. J. Rock Mech. & Min. Sci.* (1999) **36,** No. 4, 5.
37. Malinverno, A.: “A Simple Method to Estimate the Fractal Dimension of a Self Affine Series”. *Geophys. Res. Lett.* (1989) **17,** 1953.
38. Raven, K.G. and Gale, J.E.: “Water Flow in a Natural Rock Fracture as a Function of Stress and Sample Size,” *Int. J. Rock Mech. & Min. Sci.* (1985) **22,** No. 4, 251.
39. Putra, E., Hidayati, D., and Schechter, D.S.: “Reservoir Modeling of CO_2 Sequestration and Enhanced Oil Recovery in Teapot Dome,” Project Narrative, Contract No. DE-PS26-04NT42249-1D, U.S. DOE/NPTO (October 2004).
40. Lorenz, J. and Cooper, S.: “Fractures in the Core from RMOTC 48X28 Core,” Internal report to RMOTC, (June 24, 2004).
41. Davy, P., Sornette, A., and Sornette, D.: “Some Consequences of a Proposed Fractal Nature of Continental Faulting,” *Nature* (1990) **348,** 56.
42. Vicsek, T.: *Fractal Growth Phenomena,* World Scientific, Singapore (1992).
43. Schertzer, D. and Lovejoy, S.: “Physical Modeling and Analysis of Rain and Clouds by Anisotropic Scaling Multiplicative Processes,” *J. Geophys. Res.* (1987) **92**, 9693.
44. Meakin, P.: “Invasion Percolation on Substrates with Correlated Disorder,” *Physica A* (1991) **173**, 305.
45. Dreuzy, J.R. *et al.*: “Anomalous Diffusion Exponents in Continuous Two-Dimensional Multifractal Media,” *Phys. Rev. E* (2004) **70**, 016306.

46. Ozkaya, S.I., Kolkman, W., and Amthor, J.: "Mechanical Layer-Dependent Fracture Characteristics from Fracture Density vs. Tvd Cross Plots. Examples from Horizontal Wells in Carbonate Reservoirs, North Oman," paper presented at the 2003 AAPG International Conference, Barcelona, Spain, 21-24 September.
47. Vermilye, J.M. and Sholtz, C.H.: "Relation Between Vein Length and Apertures," *J. Structural Geology* (1995) **17**, No. 3, 423.
48. Voss, R.F.: "Fractals in Nature: From Characterization to Simulation," *The Science of Fractal Images*, H.-O. Peitgen and D. Saupe (eds.), Springer-Verlag, New York (1988) Chap. 1, 21-70.
49. Moltz, F.J., Liu, H.H., and Szulga, J.: "Fractional Brownian Motion and Fractional Gaussian Noise in Subsurface Hydrology: A Review, Presentation of Fundamental Properties, and Extension," *Water Resour. Res.* (1997) **33**, No. 10, 2273.
50. McGaughey, D.R. and Aitken, G.J.M.: "Statistical Analysis of Successive Random Additions for Generating Fractional Brownian Motion," *Physica A* (2000) **277**, 25.
51. Liu, H.-H. *et al.*: "A Corrected and Generalized Successive Random Additions Algorithm for Simulating Fractional Levy Motions," *Mathematical Geology* (2004) **36**, No. 3, 361.
52. Saupe, D.: "Algorithms for Random Fractals," *The Science of Fractal Images,* H.-O. Peitgen and D. Saupe (eds.), Springer-Verlag, New York (1988) Chap. 2, 71-113.
53. Bour, O. and Davy, P.: "Clustering and Size Distributions of Fault Patterns: Theory and Measurements," *Geophys. Res. Lett.* (1999) **26,** No. 13, 2001.

54. Chang, J. and Yortsos, Y.C.: "Pressure-Transient Analysis of Fractal Reservoirs," *SPEFE* (March 1990) 31; *Trans*., AIME, **289**.

55. Acuna, J.A., Ershaghi, I., and Yortsos, Y.C.: "Practical Application of Fractal Pressure-Transient Analysis in Naturally Fractured Reservoirs," *SPEFE* (September 1997) **10,** No. 3, 173.

APPENDIX A

DERIVATION OF 3D DENSITY TERM

A.1 Piggott's Model with Disc-Shaped Fractures

In Piggott's model, the number of fractures in 2D and 3D is expressed by Eq. A.1 and A.2, respectively.

$$N_l = \frac{\mathrm{a}_l}{l^{F_l}} \quad \text{(A.1)}$$

$$N_D = \frac{\mathrm{a}_D}{D^{F_D}} \quad \text{(A.2)}$$

where, *N* is the number of fractures, *l* represents length of a fracture in 2D, *D* is a fracture diameter in 3D, a_l and a_D are Piggott's density terms in 2D and 3D, and *F* is a fractal dimension which is equivalent to the fractal dimension of fracture length distribution, D_l. Thus, $F_D = F_l + 1$.

Piggott's 2D and 3D density terms are related by Eq. A.3.

$$\mathrm{a}_D = \mathrm{a}_l \sqrt{\pi}\, \frac{F_l}{F_l + 1} \frac{\Gamma\left(\frac{3+F_l}{2}\right)}{\Gamma\left(\frac{2+F_l}{2}\right)} \quad \text{(A.3)}$$

A.2 First Order Model

Equivalent equations of first order model to Piggott's model are expressed by Eq. A.4 and A.5.

$$N(L) = \frac{\alpha}{D_l} L^{D_c} {l_{\min}}^{-D_l} \quad \text{(A.4)}$$

$$N(L) = \frac{\alpha_{3D}}{D_{l3D}} L^{D_{c3D}} {l_{\min}}^{-D_{l3D}} \quad \text{(A.5)}$$

A.3 Derivation

Eq. A.1 and A.4 and Eq. A.2 and A.5 have similar forms, but are not identical due to the side length term, *L*, in the first order model. By dividing Eq. A.4 and A.5 by the side length term, two models can be equated (Eq. A.6 and A.7).

$$N_l = N(L)/L^{D_c} \quad \text{(A.6)}$$

$$N_D = N(L)/L^{D_{c3d}} \quad \text{(A.7)}$$

From Eq. A.6 and A.7, the density terms of the two models are related (Eq. A.8 and A.9).

$$\mathrm{a}_l = \frac{\alpha}{D_l} \quad \text{(A.8)}$$

$$\mathrm{a}_D = \frac{\alpha_{3D}}{D_{l3D}} \quad \text{(A.9)}$$

By substituting Eq. A.8 and A.9 for Piggott’s density terms in Eq. A.3, Eq. A.3 can be expressed with the terms of the first order model (Eq. A.10).

$$\frac{\alpha_{3d}}{D_{l3D}} = \frac{\alpha}{D_l}\sqrt{\pi}\,\frac{D_l}{D_l+1}\,\frac{\Gamma\left(\frac{3+D_l}{2}\right)}{\Gamma\left(\frac{2+D_l}{2}\right)} \quad \text{(A.10)}$$

By doing simple algebraic manipulations, the 3D density term of first order model is expressed by Eq. A.11.

$$\alpha_{3d} = \alpha\sqrt{\pi}\,\frac{\Gamma\left(\frac{3+D_l}{2}\right)}{\Gamma\left(\frac{2+D_l}{2}\right)} \quad \text{(A.11)}$$

APPENDIX B

FIELD DATA ANALYSIS AND APPLICATION

Outcrop maps have a lot of information on the geometry of fracture networks **(Fig. B-1)**. The first analysis of outcrop maps is mapping fracture center coordinates **(Fig. B-2)**. Then, these coordinates are analyzed using Eq. 4.2. Pair correlation is examined and fractal dimension of fracture center distribution is obtained **(Fig. B-3)**. The obtained fractal dimension of fracture center distribution of Bridger Gap is used as an input for generating fracture center distribution of FDFN **(Fig. B-4)**. Fractal dimension of fracture center distribution is 1.691.

The next step is analyzing the orientation analysis of the outcrop map. The needed parameters on the orientation are number of sets, mean directions of each set, and Fisher's constant of each set. Bridger Gap fractures consist of two sets of fractures, sub-vertical (set 1) and sub-horizontal (set 2). The mean orientations of set 1 and set 2 are N13.2^{o}E and N83^{o}W, respectively. Fisher's constants of set 1 and 2 calculated using Eq. 4.9. are 103 and 62.

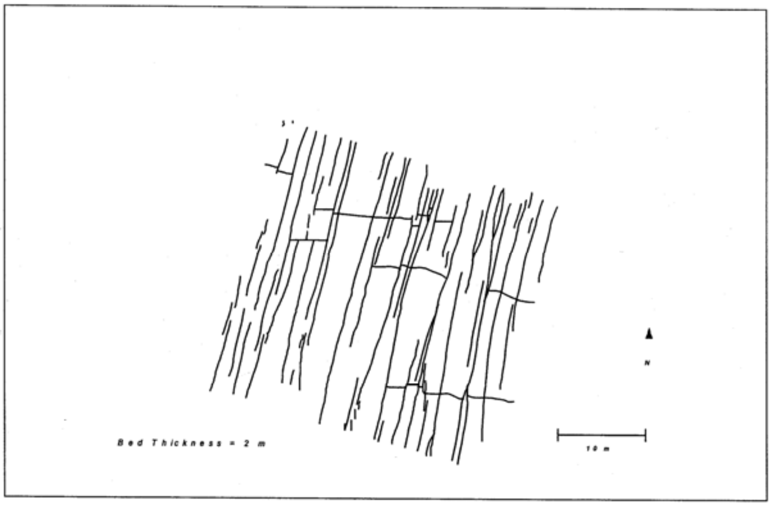


Fig. B-1-Outcrop map of Bridger Gap, Wyoming. The investigation domain size is about 40 m by 40 m.

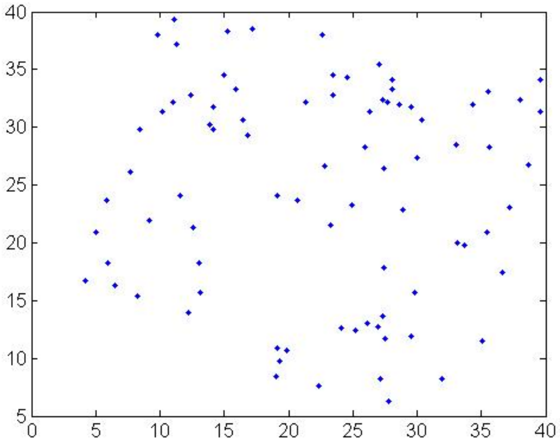


Fig. B-2-Fracture centers of the outcrop map shown in Fig. B-1.

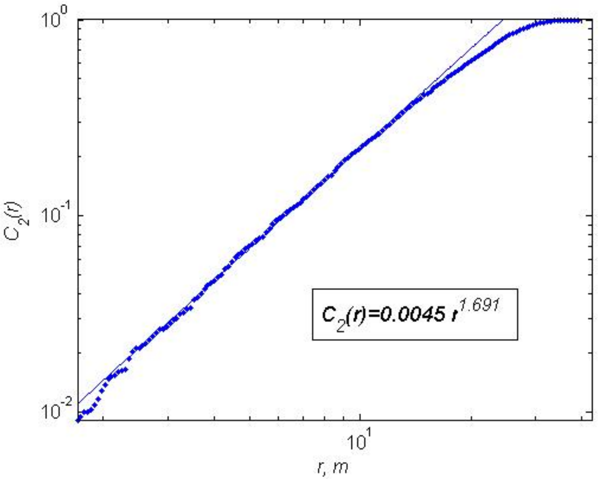


Fig. B-3-The result of pair correlation analysis. The fractal dimension of fracture center distribution of Bridger Gap is 1.691.

The following step is analyzing fracture length distribution and density using Eq. 4.1. Fig. B-5 shows the results of the analysis. Fractal dimension of fracture length distribution is 1.266 and the density term is 0.965.

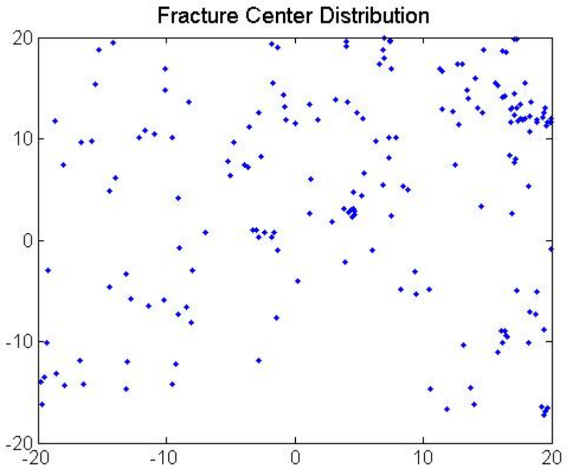


Fig. B-4-Fracture center distribution generation using the results of Bridger Gap outcrop map analysis.

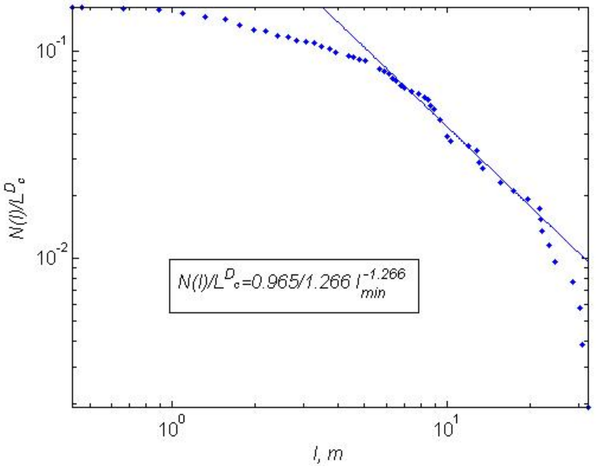


Fig. B-5-The result of density-length analysis. The fractal dimension of fracture length distribution of Bridger Gap is 1.266 and the density term is 0.965.

Through the analysis steps explained above, the necessary data to generate FDFN are obtained. These obtained data are used as input data. **Fig. B-6** is the sample FDFN generated using data from the Bridger Gap outcrop map. **Fig. B-7** compares the orientations of the outcrop map and FDFN.

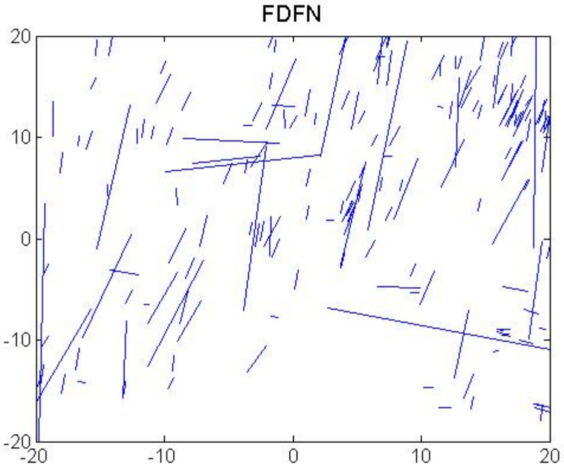
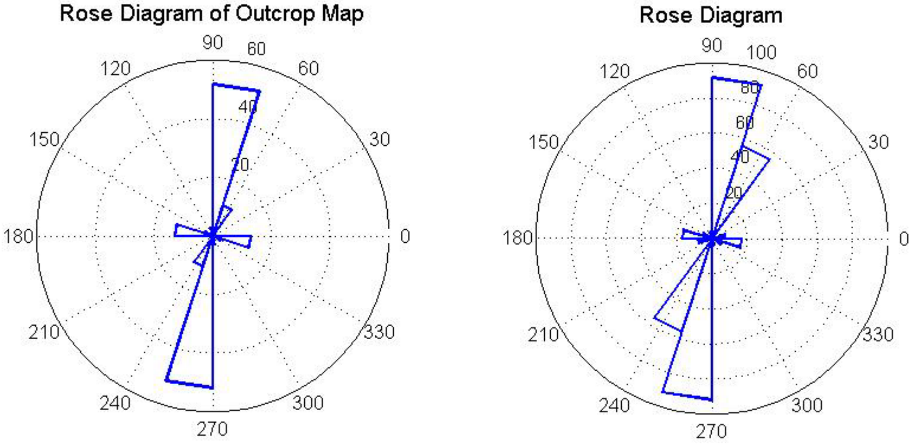


Fig. B-6-Sample FDFN generated using the data from the Bridger Gap outcrop map.



a. Bridger Gap

b. Sample FDFN

Fig. B-7-Rose diagrams of the Bridger Gap outcrop map (Fig. B-1) and the sample FDFN (Fig. B-6).

The generated FDFN shown in Fig-B-6 only have geometric information of fracture networks. To estimate fracture porosity, one more data is required. That is aperture distribution. Using the experimental procedures explained in Chapter II and III, fractal dimension of aperture distribution and amplitude can be obtained. Using the experiment results, aperture distributions are generated shown in **Fig. B-8**. Such fracture profiles are given to each fracture so that every fracture has its own aperture distribution.

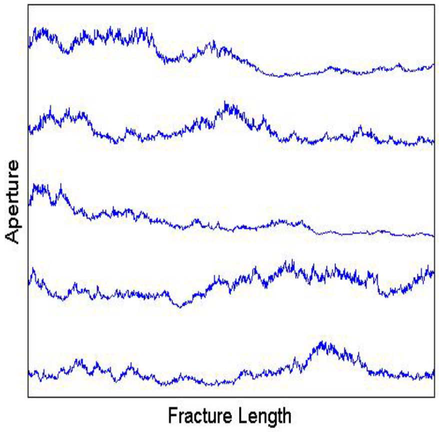


Fig. B-8-Examples of rough fracture aperture profiles. By having rough aperture distributions, FDFN have fracture volume.

APPENDIX C

2D FDFN GENERATION CODES

The codes are written with MATRAB®.

```
% This program generates 2D fractal fracture networks by means of
% Multiplicative cascade process.
% Refering the article "Connectivity properties of two-dimensional
fracture
% networks with stochastic fractal correlation.
% Input Units: [m] for DFN
%              [mm] for aperture
% Internal calculation units: All unit will be transformed to [m]
% Aperture unit is transformed to [m] after exponential is taken.

%======================================================================
%                            VARIABLES
%======================================================================
% lratio        l of the equation notation.
% FDc           Fractal dimension of fracture center distribution.
% FDl           Fractal dimension of fracture trace length distribution.
%               It is the same as (a=FDl+1)
% L             Side length of generation domain.
% alpha         Fracture density term.
% lmin          Min. length of fracture.
% nSet          Number of fracture sets. Max. values is 3.
%               If nSet = 0, then it generates random orientations.
% fracAnglei    Orientation. i = 1~3. Angle measured from the x-axis
%               in counterclockwise.
% FisherKi      Fisher constant for nSet=1~3.
% OrienProbi    Probability of each orientation. i=1~3
% q             Multifractal index
% Aden          Aperture density (ea/m)
% Amplitude     Amplitude of self affine fractal
% H             Hurst exponent
% Maxi          Maximum value of the measured data
```

```
% Mini          Minimum value of the measrued data
% STD           STD value of the measured data
% bFlag         Boundary effect flag. 1: on, 2: off

% niter         number of iterations of the process
% nFrac         Number of generated fractures.
% p()           Probability of Multiplicative cascade process.
%               if lratio=2, then 4 values will be generated.
%               if lratio=3, then 9 values will be generated.
%  Criteria       Integer  to  chech  probability  values  are  properly
generated
% ranPos()      Random position of probability
% pPar()        Probability of parent cells
% pOffs()       Probability of offspring cells
% inum          Integer indexing iteration of loops
% jnum          Integer indexing iteration of loops
% knum          Integer indexing iteration of loops
% pCellNum     Parent Cell Number
%  offIndex        indices  of  a  offspring  cell  number  of  a  reference
parent
%               cell
% ncol          Cell numbers of offspring cells which are the no. 1 in
%               their parent cells
% CellCoord     The origin coordinates of each cell.
% FracProb      Probability of fracture.
% cumProb       Cummulative probability of cells.
% fracLoc       Fracture locations according to proper probability.
% AngDev()      Angular Deviation.
% FracData1()   FracData1(a, b),
%               'a' refers fracture number
%               'b' refers data type. (fracture center coordinates).
%                   1: x coordinate, 2: y coordinate
%                   3: length,       4: orientaion
% FracData2()   FracData2(a, b), - for Extended domain
%               'a' refers fracture number
%               'b' refers data type.
%                   1: x coordinate of x1, 2: y coordinate of y1
%                   3: x coordinate of x2, 2: y coordinate of y2
```

```
% FracData3()   FracData3(a, b), - for Desired domain
%               'a' refers fracture number
%               'b' refers data type.
%                   1: x coordinate of x1, 2: y coordinate of y1
%                   3: x coordinate of x2, 2: y coordinate of y2
%                   5: Orientation
% RoseData()    Data for drawing a Rose Daigram.
% fLeng          Fracture length after trimming fractures. This values
are
%               used for aperture generation.
% Maxlevel      maximal number of recursions
% Sigma         initial standard deviation
% Xval()        Generated random aperture values
% delta()       Array holding standard deviations
% Eps           Acceptable error
% Ubound        Upper boundary of the generation
% Lbound        Lower boundary of the generation
% Range         Boolean variable deciding whether Xval are in range
%               1 : True
%               2 : False
% Outbound1     Locations of data of which data are bigger than Ubound
% Outbound2     Locations of data of which data are smaller than Lbound
% Awidth        Width of one aperture value
% fVol          Fracture Volume [m^3/m]
% fAper()       Aperture Distributioins of all fractures
% fPoro         Fracture porosity (fVol/L^2)
%======================================================================

clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Read Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%*********************************************************************
%                               DFN Data
%*********************************************************************
 lratio=input('Input Scale ratio, lratio : ', 's');
```

```
 lratio=str2double(lratio);
 FDc=input('Input Fractal dimension of center, FDc : ', 's');
 FDc=str2double(FDc);
 FDl=input('Input Fractal dimension of length, FDl : ', 's');
 FDl=str2double(FDl);
 L=input('Input Side Length of the Domain, L : ', 's');
 L=str2double(L);
 bFlag=input('Input Boundary Effect Flag, bFlag : ', 's');
 bFlag=str2double(bFlag);
 alpha=input('Input Fracture Density, alpha : ', 's');
 alpha=str2double(alpha);
 lmin=input('Input Min. Fracture Length, lmin : ', 's');
 lmin=str2double(lmin);
 q=input('Input Multifractal Index, q : ', 's');
 q=str2double(q);
% If q==1, Poisson process like results will happen.

 nSet=input('Input No. of Fracture Sets, nSet : ', 's');
 nSet=str2double(nSet);
switch nSet
    case 1
        % Unit of input angle is in 'degree'.
        fracAngle1=input('Input  Frac.  orientation,  fracAngle1  :  ', 
's');
        fracAngle1=str2double(fracAngle1);
        FisherK1=input('Input Fisher Cosnt., FisherK1 : ', 's');
        FisherK1=str2double(FisherK1);
        OrienPorb1=input('Input Porb. of Orien., OrienPorb1 : ', 's');
        OrienPorb1=str2double(OrienPorb1);
        if OrienProb1>1
            OrienProb1=1;
        end     % end of "if, OrienProb1"
    case 2
        fracAngle1=input('Input  Frac.  orientation,  fracAngle1  :  ', 
's');
        fracAngle1=str2double(fracAngle1);
        FisherK1=input('Input Fisher Cosnt., FisherK1 : ', 's');
        FisherK1=str2double(FisherK1);
```

```
        OrienProb1=input('Input Porb. of Orien., OrienProb1 : ', 's');
        OrienProb1=str2double(OrienProb1);
        fracAngle2=input('Input  Frac.  orientation,  fracAngle2  :  ',
's');
        fracAngle2=str2double(fracAngle2);
        FisherK2=input('Input Fisher Cosnt., FisherK2 : ', 's');
        FisherK2=str2double(FisherK2);
        OrienProb2=input('Input Porb. of Orien., OrienProb2 : ', 's');
        OrienProb2=str2double(OrienProb2);
        if (OrienProb1+OrienProb2)>1
            error('Probability should be less than 1')
        end     % end of "if, OrienProb1+OrienPorb2"
    case 3
        fracAngle1=input('Input  Frac.  orientation,  fracAngle1  :  ',
's');
        fracAngle1=str2double(fracAngle1);
        FisherK1=input('Input Fisher Cosnt., FisherK1 : ', 's');
        FisherK1=str2double(FisherK1);
        OrienProb1=input('Input Porb. of Orien., OrienProb1 : ', 's');
        OrienProb1=str2double(OrienProb1);

        fracAngle2=input('Input  Frac.  orientation,  fracAngle2  :  ',
's');
        fracAngle2=str2double(fracAngle2);
        FisherK2=input('Input Fisher Cosnt., FisherK2 : ', 's');
        FisherK2=str2double(FisherK2);
        OrienProb2=input('Input Porb. of Orien., OrienProb2 : ', 's');
        OrienProb2=str2double(OrienProb2);

        fracAngle3=input('Input  Frac.  orientation,  fracAngle3  :  ',
's');
        fracAngle3=str2double(fracAngle3);
        FisherK3=input('Input Fisher Cosnt., FisherK3 : ', 's');
        FisherK3=str2double(FisherK3);
        OrienProb3=input('Input Porb. of Orien., OrienProb3 : ', 's');
        OrienProb3=str2double(OrienProb3);

        if (OrienProb1+OrienProb2+OrienProb3)>1
```

```
            error('Probability should be less than 1')
        end     % end of "if, OrienProb1+OrienPorb2+OrienPorb3"
end     % end of "switch, nSet"

%**********************************************************************
%                               Aperture Data
%**********************************************************************
Aden=input('Input Aperture density (ea/m) : ', 's');
Aden=str2double(Aden);
Amplitude=input('Input Amplitude : ', 's');
Amplitude=str2double(Amplitude);
H=input('Input Hurst Exponent : ', 's');
H=str2double(H);
Maxi=input('Input Maximum value : ', 's');
Maxi=str2double(Maxi);
Mini=input('Input Minimum value : ', 's');
Mini=str2double(Mini);
STD=input('Input STD value : ', 's');
STD=str2double(STD);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          End of Read Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch bFlag
    case 1
        L=2*L;
end     % end of "switch, bFlag"

niter=round(0.5+log(L/lmin)/log(lratio));
nFrac=round(0.5+(alpha/FDl)*(L^FDc)/(lmin^FDl));
Criteria=1;
clear p;
switch q
    case 1      % Weak fractal clustering
        while Criteria==1
            p(1:(lratio^2-1))=rand(1,(lratio^2-1))./lratio;
            const=-FDc*log(lratio)-sum(p.*log(p));
            p(lratio^2)=real(const/lambertw(const));
```

```
            if sum(p.*log(p))==-FDc*log(lratio)
                Criteria=2;
                clear const
            end     % end of "if, sum(p.*log(p))"
        end     % end of "while, Criteria"
    otherwise   % Strong fractal clustering
        while Criteria==1
            p(1:(lratio^2-1))=rand(1,(lratio^2-1))./lratio^2;
            proba=(1/lratio)^((q-1)*FDc)-sum(p.^q);
            if proba>0
                Criteria=2;
                p(lratio^2)=proba^(1/q);
                clear proba
            end     % end of "if, proba"
        end     % end of "while, Criteria"
end     % end of "switch, q"
clear Criteria;

% Randomly permute probability for the first iteration
% Local cell ordering is followed:
%    1   |   3
% ---------------
%    2   |   4
ranPos=randperm(lratio^2);
pPar=p(ranPos);

% Generation of offspring cells and Assign probability
for iteration=2:niter
    pCellNum=1;
    for inum=1:lratio:(lratio)^iteration
        iline=(inum+(lratio-1))/lratio;
        ncol=(1+(iline-1)*lratio^(iteration+1):lratio:...
              lratio^iteration+(iline-1)*lratio^(iteration+1));
        switch lratio
            case 2
                offindex=[ncol, ncol+1, ncol+lratio^iteration, ...
                  ncol+lratio^iteration+1];
            case 3
```

```
                offindex=[ncol,             ncol+1,             ncol+2,
ncol+lratio^iteration, ...
                  ncol+lratio^iteration+1, ncol+lratio^iteration+2, ...
                  ncol+2*lratio^iteration,
ncol+2*lratio^iteration+1, ...
                  ncol+2*lratio^iteration+2];
        end     % end of "switch, lratio"
        offindex=reshape(offindex, lratio^(iteration-1), lratio^2);

        % Assigning prob. in y-direction (j-direction).
        % jline refers row number in 'offindex'.
        jline=1;
        for jnum=1:lratio:(lratio)^iteration
            linepos=offindex(jline,:);
            ranPos=randperm(lratio^2);
            pOffs(linepos)=pPar(pCellNum)*p(ranPos);
            pCellNum=pCellNum+1;
             jline=jline+1;
        end     % end of for "jnum"
    end     % end of for "inum"
    pPar=pOffs;
    pOffs=[];
end     % end of for "iteration"
clear pOffs; clear offindex; clear ncol; clear linepos

xCoord=(-L/2:L/lratio^niter:L/2*(1-lratio^(-niter)))';
yCoord=(-L/2:L/lratio^niter:L/2*(1-lratio^(-niter)))';

for inum=1:lratio^niter
    ncol=(1+(inum-1)*lratio^niter:inum*lratio^niter);
    % CellCoord(cell no., coordinate), coordinate :1-(x), 2-(y)
    CellCoord(ncol, 1)=xCoord;
    CellCoord(ncol, 2)=yCoord(inum);
end     % end of "for, inum"

cumProb=cumsum(pPar);
if cumProb(length(cumProb))>1
    FracProb=rand(1, nFrac).*cumProb(length(cumProb));
```

```
else
    FracProb=rand(1, nFrac);
end     % end of "if, cumProb"

clear pPar;

for inum=1:lratio^(2*niter)
    % inum refers cell number.
    switch inum
        case 1
           fracLoc=find(FracProb>0 & FracProb<=cumProb(inum));

FracData1(fracLoc,1)=CellCoord(inum,1)+rand(size(fracLoc))*...
                                (L/lratio^niter);

FracData1(fracLoc,2)=CellCoord(inum,2)+rand(size(fracLoc))*...
                                (L/lratio^niter);

FracData1(fracLoc,3)=((alpha/FDl)*(L^FDc./fracLoc)).^(1/FDl);
        otherwise
           fracLoc=find(FracProb>cumProb(inum-1) &...
                        FracProb<=cumProb(inum));

FracData1(fracLoc,1)=CellCoord(inum,1)+rand(size(fracLoc))*...
                                (L/lratio^niter);

FracData1(fracLoc,2)=CellCoord(inum,2)+rand(size(fracLoc))*...
                                (L/lratio^niter);

FracData1(fracLoc,3)=((alpha/FDl)*(L^FDc./fracLoc)).^(1/FDl);
    end     % end of "switch, inum"
end     % end of "for, inum"

row1=find(FracData1(:,3)>0);
FracData1=FracData1(row1,:);
nFrac=length(FracData1);
if cumProb(length(cumProb))>1
    FracProb=rand(1, nFrac).*cumProb(length(cumProb));
```

```
else
    FracProb=rand(1, nFrac);
end     % end of "if, cumProb"

% Assigning fracture orientations.
switch nSet
    case 0
        FracData1(:,4)=rand(nFrac,1).*pi;
    case 1
        if OrienProb1~=1
            FracProb=rand(nFrac,1);
            fracLoc=find(FracProb<=OrienProb1);
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle1+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle1-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"
            FracData1(fracLoc,4)=FracData1(fracLoc,4).*(pi/180);

            fracLoc=find(FracProb>OrienProb1 & FracProb<=1);
            FracData1(fracLoc, 4)=rand(size(fracLoc)).*pi;
        else
            AngDev=rad2deg(acos(log(1.-rand(nFrac,1))./FisherK1+1));
            AngProb=rand(nFrac,1);
            AngLoc=find(AngProb<=0.5);
            FracData1(AngLoc,4)=fracAngle1+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                FracData1(AngLoc,4)=fracAngle1-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"
            FracData1(:,4)=FracData1(:,4).*(pi/180);
        end     % end of "if, OrienProb1"
```

```
    case 2
        if (OrienProb1+OrienProb2)~=1
            FracProb=rand(1,nFrac);
            fracLoc=find(FracProb<=OrienProb1);
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle1+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle1-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"
            FracData1(fracLoc,4)=FracData1(fracLoc,4).*(pi/180);

            fracLoc=find(FracProb>OrienProb1 &...
                         FracProb<=(OrienProb1+OrienProb2));
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle2+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle2-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"
            FracData1(fracLoc,4)=FracData1(fracLoc,4).*(pi/180);

            fracLoc=find(FracProb>(OrienProb1+OrienProb2)             &
FracProb<=1);
            FracData1(fracLoc, 4)=rand(size(fracLoc)).*pi;
        else
            FracProb=rand(1,nFrac);
            fracLoc=find(FracProb<=OrienProb1);
```

```
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle1+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle1-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"

            fracLoc=find(FracProb>OrienProb1 &...
                         FracProb<=(OrienProb1+OrienProb2));
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle2+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle2-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"
            FracData1(:,4)=FracData1(:,4).*(pi/180);
        end     % end of "if, (OrienProb1+OrienProb2)"
    case 3
        if (OrienProb1+OrienProb2+OrienProb3)~=1
            FracProb=rand(1,nFrac);
            fracLoc=find(FracProb<=OrienProb1);
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle1+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
```

```
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle1-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"
            FracData1(fracLoc,4)=FracData1(fracLoc,4).*(pi/180);

            fracLoc=find(FracProb>OrienProb1 &...
                         FracProb<=(OrienProb1+OrienProb2));
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle2+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle2-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"
            FracData1(fracLoc,4)=FracData1(fracLoc,4).*(pi/180);

            fracLoc=find(FracProb>(OrienProb1+OrienProb2) &...
                         FracProb<=(OrienProb1+OrienProb2+OrienProb3));
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle3+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle3-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"
            FracData1(fracLoc,4)=FracData1(fracLoc,4).*(pi/180);

            fracLoc=find(FracProb>(OrienProb1+OrienProb2+OrienProb3)
&...
```

```
                          FracProb<=1);
            FracData1(fracLoc, 4)=rand(size(fracLoc)).*pi;
        else
            FracProb=rand(1,nFrac);
            fracLoc=find(FracProb<=OrienProb1);
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle1+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle1-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"

            fracLoc=find(FracProb>OrienProb1 &...
                          FracProb<=(OrienProb1+OrienProb2));
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
            FracData1(AngLoc2,4)=fracAngle2+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle2-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"

            fracLoc=find(FracProb>(OrienProb1+OrienProb2) &...
                          FracProb<=(OrienProb1+OrienProb2+OrienProb3));
            AngDev=rad2deg(acos(log(1.-
rand(size(fracLoc)))./FisherK1+1));
            AngProb=rand(size(fracLoc));
            AngLoc=find(AngProb<=0.5);
            AngLoc2=fracLoc(AngLoc);
```

```
            FracData1(AngLoc2,4)=fracAngle3+AngDev(AngLoc);
            AngLoc=find(AngProb>0.5);
            if length(AngLoc)>0
                AngLoc2=fracLoc(AngLoc);
                FracData1(AngLoc2,4)=fracAngle3-AngDev(AngLoc);
            end     % end of if, "length(AngLoc)>0"
            FracData1(:,4)=FracData1(:,4).*(pi/180);
        end     % end of "if, (OrienProb1+OrienProb2+OrienProb3)"
end     % end of "switch, nSet"
clear AngDev; clear AngProb; clear AngLoc; clear AngLco2;

% Calculate end points of fractures.
FracData2(:,1)=FracData1(:,1)+FracData1(:,3)./2.*cos(FracData1(:,4));
FracData2(:,2)=FracData1(:,2)+FracData1(:,3)./2.*sin(FracData1(:,4));
FracData2(:,3)=FracData1(:,1)-FracData1(:,3)./2.*cos(FracData1(:,4));
FracData2(:,4)=FracData1(:,2)-FracData1(:,3)./2.*sin(FracData1(:,4));

switch bFlag
    case 1
%**********************************************************************
%                 Added for considering the Boundary Effect
%**********************************************************************
% Cut the sampling domain from the generation domain
        L=L/2;

        FracData3=FracData2;
% Identify x1 & x2 are greater than L/2
        TrimIndex=find(FracData3(:,1)>L/2 & FracData3(:,3)>L/2);
        FracData3(TrimIndex,:)=100*L;
% Identify x1 & x2 are less than -L/2
        TrimIndex=find(FracData3(:,1)<-L/2 & FracData3(:,3)<-L/2);
        FracData3(TrimIndex,:)=100*L;
% Identify y1 & y2 are greater than L/2
        TrimIndex=find(FracData3(:,2)>L/2 & FracData3(:,4)>L/2);
        FracData3(TrimIndex,:)=100*L;
% Identify y1 & y2 are less than -L/2
        TrimIndex=find(FracData3(:,2)<-L/2 & FracData3(:,2)<-L/2);
        FracData3(TrimIndex,:)=100*L;
```

```
%**********************************************************************
    otherwise
        FracData3=FracData2;
        clear FracData2;
end     % end of "switch, bFlag"

% Trim fracture of which ends are out of the domain.
% x1 & x2 > L/2
TrimIndex=find(FracData3(:,1)>L/2 & FracData3(:,1)~=100*L);
FracData3(TrimIndex,1)=L/2;
FracData3(TrimIndex,2)=tan(FracData1(TrimIndex,4)).*L/2+...
                (FracData1(TrimIndex,2)-tan(FracData1(TrimIndex,4)).*...
                FracData1(TrimIndex,1));
TrimIndex=find(FracData3(:,3)>L/2 & FracData3(:,3)~=100*L);
FracData3(TrimIndex,3)=L/2;
FracData3(TrimIndex,4)=tan(FracData1(TrimIndex,4)).*L/2+...
                (FracData1(TrimIndex,2)-tan(FracData1(TrimIndex,4)).*...
                FracData1(TrimIndex,1));
% x1 & x2 < -L/2
TrimIndex=find(FracData3(:,1)<-L/2);
FracData3(TrimIndex,1)=-L/2;
FracData3(TrimIndex,2)=tan(FracData1(TrimIndex,4)).*(-L/2)+...
                (FracData1(TrimIndex,2)-tan(FracData1(TrimIndex,4)).*...
                FracData1(TrimIndex,1));
TrimIndex=find(FracData3(:,3)<-L/2);
FracData3(TrimIndex,3)=-L/2;
FracData3(TrimIndex,4)=tan(FracData1(TrimIndex,4)).*(-L/2)+...
                (FracData1(TrimIndex,2)-tan(FracData1(TrimIndex,4)).*...
                FracData1(TrimIndex,1));
% y1 & y2 > L/2
TrimIndex=find(FracData3(:,2)>L/2 & FracData3(:,2)~=100*L);
FracData3(TrimIndex,2)=L/2;
FracData3(TrimIndex,1)=(L/2-FracData1(TrimIndex,2)+...

tan(FracData1(TrimIndex,4)).*FracData1(TrimIndex,1))./...
                tan(FracData1(TrimIndex,4));
TrimIndex=find(FracData3(:,4)>L/2 & FracData3(:,4)~=100*L);
FracData3(TrimIndex,4)=L/2;
```

```
FracData3(TrimIndex,3)=(L/2-FracData1(TrimIndex,2)+...

tan(FracData1(TrimIndex,4)).*FracData1(TrimIndex,1))./...
                tan(FracData1(TrimIndex,4));
% y1 & y2 < -L/2
TrimIndex=find(FracData3(:,2)<-L/2);
FracData3(TrimIndex,2)=-L/2;
FracData3(TrimIndex,1)=(-L/2-FracData1(TrimIndex,2)+...

tan(FracData1(TrimIndex,4)).*FracData1(TrimIndex,1))./...
                tan(FracData1(TrimIndex,4));
TrimIndex=find(FracData3(:,4)<-L/2);
FracData3(TrimIndex,4)=-L/2;
FracData3(TrimIndex,3)=(-L/2-FracData1(TrimIndex,2)+...

tan(FracData1(TrimIndex,4)).*FracData1(TrimIndex,1))./...
                tan(FracData1(TrimIndex,4));

switch bFlag
    case 1
        % Identify x1 & x2 are greater than L/2
        TrimIndex=find(FracData3(:,1)>L/2 & FracData3(:,3)>L/2);
        FracData3(TrimIndex,:)=100*L;
        % Identify x1 & x2 are less than -L/2
        TrimIndex=find(FracData3(:,1)<-L/2 & FracData3(:,3)<-L/2);
        FracData3(TrimIndex,:)=100*L;
        % Identify y1 & y2 are greater than L/2
        TrimIndex=find(FracData3(:,2)>L/2 & FracData3(:,4)>L/2);
        FracData3(TrimIndex,:)=100*L;
        % Identify y1 & y2 are less than -L/2
        TrimIndex=find(FracData3(:,2)<-L/2 & FracData3(:,2)<-L/2);
        FracData3(TrimIndex,:)=100*L;

        % Remove fractures which are not in the domain
        TrimIndex=find(FracData3(:,1)~= 100*L);
        FracData3=FracData3(TrimIndex,:);
        FracData3(:,5)=FracData1(TrimIndex,4);
        [nFrac bbb]=size(FracData3);
```

```
        clear bbb;
    otherwise
        FracData3(:,5)=FracData1(:,4);
end     % end of "switch, bFlag"

% Recalculate fracture length within the domain.
xLeng=FracData3(:,1)-FracData3(:,3);
yLeng=FracData3(:,2)-FracData3(:,4);
fLeng=sqrt(xLeng.^2+yLeng.^2);
clear xLeng; clear yLeng;

% Draw Fracture traces using hold function.
figure(1);
Xcoords=[FracData3(1,1), FracData3(1,3)];
Ycoords=[FracData3(1,2), FracData3(1,4)];
plot(Xcoords, Ycoords);
hold on;
for inum=2:nFrac
    Xcoords=[FracData3(inum,1), FracData3(inum,3)];
    Ycoords=[FracData3(inum,2), FracData3(inum,4)];
    plot(Xcoords, Ycoords);
end     % end of "for, inum"
hold off;
title('Fractal Discrete Fracture Networks');
axis([-L/2 L/2 -L/2 L/2]);
xlabel('L = ');
ylabel('L = ');

RoseData=[FracData3(:,5); FracData3(:,5)+pi];
figure(2);
rose(RoseData);
title('Rose Diagram of FDFN');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Aperture Calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for iAper=1:nFrac
```

```
Maxlevel=round(0.5+log(Aden*fLeng(iAper))/log(2));
if Maxlevel<1
    Maxlevel=1;
end     % end of "if, Maxlevel"
Sigma=log(Amplitude/(3.86*10^(-4)*(2-H)^10.42));
Eps=10^-6;
N=2^Maxlevel;
Awidth=fLeng(iAper)/(N+1);
Xval=zeros(N+1, 1);
delta=zeros(Maxlevel,1);
index=(1:Maxlevel)';
Ubound=log(Maxi+STD*3);

if (Mini-STD)<=0
    Lbound=log(Mini);
elseif (Mini-STD*2)<=0
    Lbound=log(Mini-STD);
elseif (Mini-STD*3)<=0
    Lbound=log(Mini-STD*2);
else
    Lbound=log(Mini-STD*3);
end     % end of "if, (Mini-STD)"

delta=Sigma*0.5.^(index.*H)*sqrt(0.5)*sqrt(1-2^(2*H-2));

Range=2;
while Range==2
    Xval(N+1)=Sigma*randn;
    if Xval(N+1)<=Ubound & Xval(N+1)>= Lbound
        Range=1;
    end     % end of "if, Xval(N+1)"
end     % end of "while, Range"

D=N;
d=N/2;
for level=1:Maxlevel
    index=(d:D:N-d)';
    index=index+1;
```

```
        indexL=index-d;
        indexR=index+d;
        Xval(index)=0.5.*(Xval(indexL)+Xval(indexR));

        index=(0:d:N)';
        index=index+1;
        Xval(index)=Xval(index)+delta(level).*randn(size(index));
        Range=2;
        while Range==2
            Outbound=find(Xval<Lbound | Xval>Ubound);
            if length(Outbound)==0
                Range=1;
            else
                Xvalnew=zeros(size(Xval))+100*Ubound;
                Xvalnew(Outbound)=Xval(Outbound)...
                           +delta(level).*randn(size(Outbound));
                Inbound=find(Xvalnew>Lbound & Xvalnew<Ubound);
                Xval(Inbound)=Xvalnew(Inbound);
                clear Xvalnew;
            end     % end of "if, length(Outbound)==0"
        end     % end of "while, Range==2"
        D=D/2;
        d=d/2;
    end     % end of "for, level"

% Correct and generalize SRA algorithm
    D=2;
    d=1;
    delta_new=delta(Maxlevel);
    while (delta_new/Sigma) >= Eps
        delta_new=delta_new*0.5^(0.5*H);
        Xval=Xval+delta_new.*randn(size(Xval));
    end     % end of "while, delta_new/Sigma"
    Xval=exp(Xval);
    Xval=Xval./1000;
    fVol(iAper)=sum(Xval.*Awidth);
    lXval=length(Xval);
    fAper(1:lXval,iAper)=Xval;
```

```
end     % end of "for, iAper"
fPoro=sum(fVol)/L^2*100;
```

VITA

Name: Tae Hyung Kim

Place of Birth: Seoul, Republic of Korea

Permanent Address: Dong-Ah Green Apt. 103-1905
Ichon 2nd dong, Yongsan-Gu
Seoul, 140-773
Republic of Korea

Education: B.S., Energy & Mineral Engineering (Feb. 2000)
Inha University, Incheon, Korea.

M.S., Geomechanical Engineering (Feb. 2002)
Seoul National University, Seoul, Korea.

Ph.D., Petroleum Engineering (Dec. 2007)
Texas A&M University, College Station, Texas, USA.

Professional Affiliations: Society of Petroleum Engineers, Member.